Nama : Anak Agung Indi Kusuma Putra

NIM : 2205551079

Kelompok : 23

|  |
| --- |
| MODUL IV STRUKTUR DAN OPERASI FILE |

**Tujuan**

1. Mengetahui dan memahami tentang *struct* serta kegunaan dari *struct*
2. Mengetahui dan memahami perbedaan antara *struct* yang bersifat *private* dan *public*
3. Mengetahui dan memahami penggunaan *struct* baik secara *private* maupun *public* dalam C/C++
4. Mengetahui dan memahami apa itu *union* dan perbedaan *union* dengan *struct*
5. Mengetahui dan memahami apa itu akses *modifier* yang ada dalam C/C++
6. Mengetahui dan memahami penggunaan tipe data *string* beserta dapat menerapkan fungsi – fungsi yang dapat di pakai untuk mengkonversi tipe data *string* ke tipe data yang lain
7. Mengetahui dan memahami penggunaan tipe data FILEdan *statement* lain yang digunakan dalam operasi *file* dalam C++.

**Tugas Pendahuluan**

1. Jelaskan tentang pengertian serta kegunaan *struct*
2. Apakah perbedaan antara *struct* yang bersifat *private* dan *public*?
3. Berikan contoh penggunaan *struct* secara *private* maupun *public* dalam C/C++
4. Jelaskan apa yang dimaksud dengan *union* serta jelaskan perbedaan *union* dengan *struct*!
5. Jelaskan macam macam akses *modifier* yang ada dalam C/C++!
6. Jelaskan tentang penggunaan tipe data *string* beserta dengan fungsi-fungsi yang dapat di pakai untuk mengkonversi tipe data *string* ke tipe data lain.
7. Jelaskan dan berikan contoh penggunaan FILE dan *statement* lain yang digunakan dalam operasi *file,* dalam C/C++.
8. Buatlah *flowchart* dan *pseudocode* (notasi algoritma) dari masing-masing soal dibawah, lalu buatlah *trace* dari *flowchart* yang anda buat!

**Jawaban**

1. **Struct**

*Struct* adalah tipe data yang didalamnya berisi kumpulan variabel yang berisi data. tipe data struct mirip dengan *array* tetapi *struct* bisa memiliki lebih dari 1 variabel . tipe data *struct* biasanya digunakan untuk menyimpan sekelompok data yang terdiri atas banyak data yang berkaitan dan ditampung dalam 1 variabel.

1. **Perbedaan Struct yang Bersifat Public dan Private**

*Struct* yang bersifat *public* berarti *struct* bisa diakses oleh kelas-kelas turunannya maupun terhadap lingkungan luar sedangkan *struct* yang bersifat *private* hanya bisa diakses oleh kelas yang diberikan hak akses. kelas yang tidak diberikan hak akses tidak akan bisa mengakses program tersebut. Untuk membuat *struct* bersifat *private.* Kita bisa menggunakan *keyword private*.

1. **Penggunaan Struct yang Bersifat Public dan Private**

Penggunaan *struct* yang bersifat *public* dan *private* hanya berbeda pada saat pendeklarasiannya saja. Dimana *struct* yang bersifat *private* akan ditambahkan dengan *keyword* *private*, begitu pula dengan *struct* yang bersifat *public* bisa dengan menambahkan *keyword public*. Contoh penggunaan *struct* bisa dilihat pada program dibawah.

|  |
| --- |
| #include <iostream.h>;  using namespace std;    class identitas{  private:      char nama[30];      char umur[2];  public:      dosen(char \*nama2, char \*umur){          Strcpy (nama, nama2);          Strcpy (umur, umur2);      }        void display(){          cout<<nama<<endl;          cout<<umur<<endl;      }  } dosen;  int main(){      dosen.identitas(“mikel”, “31”);        dosen.tampilkan();  } |

Kode Program 2.1 Contoh Penggunaan *Struct* yang Bersifat *Public* dan *Private*

Kode program 2.1 diatas merupakan contoh penggunaan *struct* yang bersifat *public* dan *private*. Variabel nama dan umur dideklarasikan secara *private* yang artinya tidak akan diakses dari data manapun sedangakan *void* dosen() dandisplay() dideklarasikan secara publik agar dapat diakses oleh data yang lain.

1. **Union**

*Union* adalah kumpulan data yang menyimpan banyak variabel yang mampu menyimpan data dengan tipe data yang berbeda. Data dalam *union* suatu memori yang sama yang membuat data saling timpa dan saling mempengaruhi satu sama lain. Jadi jika satu data dalam *union* diubah, nilai data lain dalam *union* yang sama juga ikut berubah walupun tidak menjadi sama

1. **Akses Modifier**

Akses *modifier* merupakan fitur untuk membatasi hak akses ke suatu anggota dari *class* supaya tidak sembarang perintah dapat mengakses atribut yang dibatasi aksesnya.

* 1. **Public**

*Public* merupakan akses *modifier* yang memperbolehkan anggota dari suatu *class* untuk digunakan oleh seluruh *class* yang berbeda, *class* dari *package* yang berbeda, *class* itu sendiri, *package* yang sama. Intinya siapapun bisa mengakses anggota *class* yang bersifat *public.*

* 1. **Private**

*Private* merupakan akses *modifier* yang membatasi hak akses untuk anggota *class* dimana anggota *class* hanya bisa diakses oleh *class* yang sama. Jadi *class* yang berbeda maupun *package* yang sama tidak bisa mengakses *class* yang bersifat *private.*

* 1. **Protected**

*Protected* merupakan akses *modifier* yang membatasi hak akses untuk anggota *class* dimana anggota *class* hanya bisa diakses oleh *package* yang sama. Siapapun bisa mengakses anggota *class* yang bersifat *protected*  asal berada pada *package* yang sama.

1. **Penggunaan Tipe Data String**

*String* merupakan tipe data yang menampung kumpulan karakter. Dalam C/C++ terdapat beberapa fungsi yang bisa digunakan untuk mengkonversi *string* menjadi tipe data lain. Untuk menggunakan fungsi-fungsi tersebut harus melakukan deklarasi *library* string.h.

* 1. **Konversi Tipe Data String**

Konversi tipe data string ke tipe data lain dapat dilakukan dengan menggunakan fungsi atoi(), atof(), atol(), atoll() yang terdapat pada *library* string.h *.* berikut adalah contoh dari penggunaan masing-masing fungsi diatas.

* + 1. **Fungsi Atoi()**

Fungsi atoi() digunakan untuk mengkonversikan tipe data *string* ke tipe data *integer.* Fungsi ini berada pada header stdlib.h dan hanya bisa mengkonversi data bilangan bulat. Jadi bilangan desimal dan bilangan eksponen tidak bisa dikonversi menggunakan fungsi ini. contoh dari penerapan fungsi atoi() bisa dilihat pada program dibawah.

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  int main()  {  char huruf[]= "57236";  int a = atoi(huruf);  printf("%d",a);  return 0;  } |

Kode Program 6.1.1 Contoh Penerapan Fungsi Atoi()

Kode Program 6.1.1 diatas merupakan contoh dari penerapan fungsi atoi(). Pada program diatas fungsi atoi() akan mengambil nilai *string* yang berada pada variabel huruf dan akan diubah ke tipe data *integer* lalu disimpan ke variabel a.

* + 1. **Fungsi Atof()**

Fungsi atof() merupakan fungsi pada *library* stdlib.h yang akan mengkonversi tipe data *string* menjadi tipe data *float*. Fungsi ini akan meminta argumen berupa *string* yang akan dikonversi menjadi *float*. Contoh penerapan dari fungsi atof() bisa dilihat pada program dibawah.

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  int main()  {  char phi[]= "3.1415926535";  float a = atoi(phi);  printf("%f",a);  return 0;  } |

Kode Program 6.1.2 Contoh Penerapan Fungsi Atof()

Kode program 6.1.2 merupakan contoh dari penerapan fungsi atof(). Pada program diatas nilai *string* yang ada pada variabel phi akan dikonversikan menjadi *float* oleh fungsi atof() lalu akan disimpan pada variabel a.

* + 1. **Fungsi Atol()**

Fungsi atol() merupakan fungsi pada *library* stdlib.h yang akan mengkonversi tipe data *string* menjadi tipe data *long* *integer*. Memanggil fungsi ini memerlukan argumen berupa *string* yang akan dikonversi menjadi *long* *integer*. Contoh penerapan dari fungsi atol() bisa dilihat pada program dibawah.

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  int main()  {  char string[]= "36436265889878735";  long l = atol(string);  printf("%ld",l);  return 0;  } |

Kode Program 6.1.3 Contoh Penerapan Fungsi Atol()

Kode program 6.1.3 merupakan contoh penerapan dari fungsi atol(). Program diatas mengkonversi *string* yang ada pada variabel string menggunakan fungsi atol() dan hasilnya yang berupa *long* *integer* akan disimpan pada variabel l.

1. **Penggunaan FILE dan Statement Lain Dalam Operasi File**

Penggunaan FILE pada bahasa C merupakan operasi yang sangat penting untuk dikuasai oleh *programmer*. Karena dalam pembuatan program yang sesungguhnya tentu kita membutuhkan file sebagai tempat untuk menampung data-data selama operasi program karena jika kita menggunakan variabel biasa, data yang di operasikan biasanya hanya bersifat sementara. *Statement* untuk menggunakan *file* pada bahasa C ada 3 yang utama yaitu *read*, *write*, dan *append*.

* 1. **READ**

Operasi *read* memiliki fungsi seperti namanya, yaitu untuk membuka *file* dan membaca isi *file* yang dibuka. *File* yang dibuka juga bisa disimpan pada variabel untuk memudahkan kita saat ingin mengolah isi yang ada didalamnya. Contoh sintaks untuk membuka lalu membaca file bisa dilihat contoh dibawah.

|  |
| --- |
| fopen(“file.txt”,”r”); |

Kode Program 7.1 Contoh Sintaks Untuk Membaca File

Kode program 7.1 merupakan contoh operasi untuk membuka *file* dalam Bahasa C. kita menggunakan fopen() dan berisi argumen berupa nama *file* yang akan dibuka dan mode bagaimana kita akan menggunakan *file* tersebut yang di program diatas berisi “r” karena kita hanya membuka dan membaca *file*.

* 1. **Write**

Operasi *write* memiliki arti kita membuka *file* dalam mode *write* yang berarti kita akan menulis ke file yang kita buka. Jika pada file yang dibuka sudah terdapat data di dalamnya maka data tersebut akan di *overwrite* atau ditimpa dengan *output* program. Contoh sintaks untuk operasi *write* pada bahasa C bisa dilihat pada contoh dibawah.

|  |
| --- |
| fopen(“file.txt”,”w”); |

Kode Program 7.2 Contoh Sintaks Untuk Operasi *Write*

Kode Program 7.2 berisi contoh penggunaan operasi *write* dalam bahasa C. Pertama program akan membuka *file* menggunakan fopen() yang didalamnya kita masukan argumen berupa nama *file* yang akan dibuka dan mode “w” yang berarti kita sedang membuka *file* dalam mode *write*. Jadi sebenarnya yang membedakan dari operasi yang lain hanyalah modenya saja.

* 1. **Append**

Operasi *append* memiliki fungsi untuk membuka *file* dalam mode *append* yang artinya kita bisa menambahkan data baru pada sebuah *file*. Yang membedakan operasi *append* dengan operasi *write* adalah operasi *append* tidak akan menimpa data yang sudah ada. Jadi data baru akan ditambahkan ke *file* tanpa menghilangkan data yang lama. Contoh sintaks untuk operasi append pada bahasa C bisa dilihat pada contoh dibawah.

|  |
| --- |
| fopen(“file.txt”,”a”); |

Kode Program 7.3 Contoh Sintaks Untuk Operasi *Append*

Kode Program 7.3 berisi contoh untuk menggunakan operasi append yang dimana kita akan membuka file dalam mode “a” yang berarti mode append dan bisa menambahkan data tanpa menimpa data yang memang sudah ada pada file yang kita buka.

## Flowchart, Pseudocode, dan Trace Soal Praktikum Modul IV

*Flowchart* adalah suatu representasi proses dengan menggunakan grafik atau simbol-simbol tertentu. Setiap proses di dalam *flowchart* diwakili oleh simbol-simbol yang berbeda dan mengandung deskripsi singkat di dalamnya. Simbol *flowchart* tersebut dihubungkan dengan tanda panah, menandakan proses dari alur tersebut.

*Pseudocode* merupakan suatu ‘bahasa pemrograman informal’ yang digunakan pada proses desain suatu program. *Pseudocode* memiliki kemiripan dengan kode program asli, sehingga memudahkan *programmer* dalam membayangkan implementasi suatu algoritma ke dalam kode program.

*Trace* adalah tahapan pengecekan suatu alur algoritma yang dilakukan untuk memeriksa apakah alur yang disusun memiliki kesalahan atau tidak. *Trace* dilakukan dengan cara memeriksa *input,* proses, dan *output* dari algoritma tersebut. Berikut ini adalah *flowchart, pseudocode,* serta hasil *trace* dari soal praktikum Modul IV.

### Flowchart, Pseudocode, dan Trace Program Gaji Harian

Program gaji harian merupakan program yang dapat menghitung gaji harian dengan memanfaatkan *struct*. Berdasarkan ketentuan yang diberikan, program akan diberikan suatu *input* berupa jumlah jam kerja yang nantinya akan diproses oleh program sehingga program mampu memberikan *output* berupa rincian dari perhitungan gaji harian tersebut. Berikut adalah *flowchart, pseudocode,* serta *trace* dari program gaji harian.

#### Flowchart Program Gaji Harian

Program gaji memiliki *flowchart* yang menggambarkan langkah-langkah atau proses dari program dengan menggunakan simbol yang dapat dipahami oleh orang awam.

|  |
| --- |
|  |

Gambar 8.1 *Flowchart* Subproses *Input* Jam Kerja

Gambar 8.1 adalah gambaran dari fungsi meng-*input* jam kerja karyawan. Dengan mendeklarasikan beberapa variabel lalu meng-*input* jam kerja dan pilihan untuk menanyakan apakah jam yang di-*input* sudah benar atau belum. Setelah meng-*input* pilihan maka jika *input* adalah 1 maka akan di tentukan lagi apakah jam kerja yang di-*input* melebihi maksimal jam bekerja, jika tidak maka akan dibawa ke fungsi menghitung gaji utama tetapi jika melebihi maka akan dibawa ke fungsi menghitung gaji lembur. *Input* pilihan yang dimasukkan 2 maka akan dibawa kembali ke fungsi untuk memasukkan jam kerja, sama halnya jika memasukkan angka lebih dari 2 akan dibawa ke fungsi memasukkan jam kerja juga.

|  |
| --- |
|  |

Gambar 8.2 *Flowchart* Subproses Menghitung Gaji Utama

Pada Gambar 8.2 adalah prosedur untuk menghitung gaji utama jika jam kerja yang diinput tidak melebihi jam kerja maksimal. Dengan mendeklarasikan 3 variabel yaitu *float gaji\_perjam*, *float gaji\_utama*, *float jam\_kerja* lalu memproses perhitungan dengan rumus yang dipanggil menggunakan *struct h* yaitu *h.gaji\_utama = h.jam\_kerja \* gaji\_perjam*. Output dari hasil perhitungan bisa dipanggil dengan variabel *h.gaji\_utama*. Setelah proses selesai maka akan memanggil prosedur *ulang()* untuk memilih akan mengulang program atau menghentikan program.

|  |
| --- |
|  |

Gambar 8.3 *Flowchart* Subproses Menghitung Gaji Lembur

Gambar 8.3 adalah *flowchart* untuk prosedur menghitung gaji lembur jika jam kerja melebihi jam kerja maksimal. Program dimulai dengan mengdeklarasikan variabel *int jam\_bekerja*, *float gaji\_perjam*, *float gaji\_utama*, *float jam\_kerja*, *float gaji\_lembur*, *float gaji\_total*, dan *float jam\_lembur*. Setelah melakukan deklarasi, langsung akan dieksekusi dengan rumus yang dipanggil menggunakan *struct h*. Rumus untuk menghitung gaji lembur, yaitu pertama *h.jam\_lembur = h.jam\_kerja – jam\_bekerja*, kedua *h.gaji\_lembur = h.jam\_lembur \* jam\_bekerja \* gaji\_perjam*, ketiga *h.gaji\_utama = jam\_bekerja \* gaji\_perjam*, keempat *h.gaji\_total = h.gaji\_lembur + h.gaji\_utama*. Setelah semua rumus dijalankan lalu hasil akan disimpan pada variabel *h.gaji\_total* dan bisa dioutputkan jika memanggil variabel *h.gaji\_total*. Saat semua proses telah selesai maka akan memanggil prosedur *ulang()* untuk memilih akan mengulang program atau menghentikan program.

#### Pseudocode Program Gaji Harian

*Pseudocode* adalah notasi algoritma yang menyerupai kode program tetapi sintaksnya menggunakan bahasa manusia sehingga bisa dibaca oleh orang-orang awam. Bahasa manusia yang lebih sering digunakan adalah bahasa inggris. Berikut adalah *pseudocode* dari program menghitung gaji.

|  |
| --- |
| DESKRIPSI :  PROGRAM MENGHITUNG GAJI  DEKLARASI  BEGIN:  jam\_bekerja = 8 : int  gaji\_perjam = 10625 : float  struct hitung{  gaji\_lembur : float  gaji\_utama : float  gaji\_total : float  jam\_kerja : float  jam\_lembur : float  }  struct hitung h;  FUNCTION perhitungan\_gaji()  FUNCTION perhitungan\_gajiUtama()  FUNCTION perhitungan\_gajiLembur()  FUNCTION valid\_int(\*var : int) : void  DEKLARASI  BEGIN :  VAR buff[20] : char  VAR cek : char  fflush(stdin)  if(fgets(buff, sizeof(buff), stdin)!=NULL)THEN  if(sscanf(buff, "%d %c", var, &cek)==1)THEN  return 1  END  ENDIF  return 0  END  FUNCTION valid\_float(\*var : float) : float  DEKLARASI  BEGIN :  VAR buff[20] : char  VAR cek : char  fflush(stdin)  if(fgets(buff, sizeof(buff), stdin)!= NULL)  THEN  if(sscanf(buff,"%f %c",var,&cek)== 1)  THEN  return 1  END  ENDIF  return 0  END  FUNCTION input\_jamKerja(\*var : float, \*prompt : char) : void  DEKLARASI  BEGIN :  while (1)THEN  WRITE(prompt)  if(valid\_float(var))THEN  break  WRITE("\t=================================\n")  WRITE("\n !Masukan jam saja, jangan yang lain!\n")  WRITE("\n\t=================================")  WRITE("\n")  ENDWHILE  END  FUNCTION input\_pilihan(\*var : int, \*prompt : char) : void  DEKLARASI  BEGIN :  while (1)THEN  WRITE(prompt)  if(valid\_int(var)) THEN  break  WRITE("\nSudah benar atau belum?!\n")  WRITE("\n")  ENDWHILE  END  FUNCTION input\_ulang(int \*var, char \*prompt) : void  DEKLARASI  BEGIN :  while (1)THEN  WRITE(prompt)  if(valid\_int(var))  break  WRITE("\nMau mengulang atau tidak?!\n")  ENDWHILE  END  FUNCTION ulang() : void  DEKLARASI  BEGIN :  Mengulang : int  CALL input\_ulang(&mengulang, "\nIngin mengulang program?\n1. YA \n2. TIDAK\n""=> ")  switch(mengulang)  case 1:  system("cls")  CALL perhitungan\_gaji()  break  case 2:  WRITE("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TERIMA KASIH \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")  exit(0)  default:  WRITE("\nMau mengulang atau tidak?\n")  CALL ulang()  break  ENDSWITCH  END  FUNCTION perhitungan\_gaji() : void  DEKLARASI  BEGIN :  Pilihan : int  WRITE("\n\t=================================\n")  WRITE("\t||\t JAM KERJA ||\n")  WRITE("\t=================================\n")  WRITE ("\t| Masukan lama anda bekerja |\n")  WRITE("\t=================================\n")  CALL input\_jamKerja(&h.jam\_kerja, "\t| => ")  WRITE("\t=================================\n")  CALL input\_pilihan (&pilihan, "\nApakah jam kerja sudah benar?\n1. Ya \n2. Salah\n""=> ")  if (pilihan == 1)THEN  if (h.jam\_kerja<=jam\_bekerja)THEN  CALL perhitungan\_gajiUtama()  else if(h.jam\_kerja>jam\_bekerja)THEN  CALL perhitungan\_gajiLembur()  END  else if (pilihan == 2) THEN  CALL perhitungan\_gaji()  else if (pilihan >= 3)THEN  system("cls")  CALL perhitungan\_gaji()  ENDIF  system("cls")  END  FUNCTION perhitungan\_gajiUtama() : void  DEKLARASI  BEGIN :  h.gaji\_utama = h.jam\_kerja \* gaji\_perjam  system("cls")  WRITE("\n\t===========================================\n")  WRITE("\t||\t RINCIAN GAJI ||\n")  WRITE("\t===========================================\n")  WRITE("\t| Jam Kerja \t\t= %.f Jam |\n", h.jam\_kerja)  WRITE("\t| Lama anda lembur \t= 0 Jam |\n")  WRITE("\t===========================================")  WRITE("\n\t| Gaji Utama\t= %.f x Rp. 10,625 |", h.jam\_kerja)  WRITE("\n\t|\t\t= Rp. %.1f |\n", h.gaji\_utama)  WRITE("\t===========================================")  WRITE("\n\t|| GAJI ANDA\t: Rp. %.lf ||\n", h.gaji\_utama)  WRITE("\t===========================================\n")  CALL ulang()  END  FUNCTION perhitungan\_gajiLembur() : void  DEKLARASI  BEGIN :  h.jam\_lembur = h.jam\_kerja - jam\_bekerja  h.gaji\_lembur = h.jam\_lembur \* jam\_bekerja \* gaji\_perjam  h.gaji\_utama = jam\_bekerja \* gaji\_perjam  h.gaji\_total = h.gaji\_lembur + h.gaji\_utama  system("cls")  WRITE("\n\t=====================================================\n")  WRITE("\t||\t\t RINCIAN GAJI ||\n")  WRITE("\t=====================================================\n")  WRITE("\t| Maks. jam kerja/hari \t\t= %d Jam |\n", jam\_bekerja)  WRITE("\t| Jam Kerja \t\t\t= %.f Jam |\n", h.jam\_kerja)  WRITE("\t| Lembur \t\t\t= %.f jam |\n", h.jam\_lembur)  WRITE("\t=====================================================\n")  WRITE("\t| Gaji utama \t= 8 x Rp. 10,625 |\n")  WRITE("\t| \t\t= Rp. %.1f |", h.gaji\_utama)  WRITE("\n\t| |")  WRITE("\n\t| Gaji lembur \t= %.f x 8 x Rp. 10,625 |\n", h.jam\_lembur)  WRITE("\t| \t\t= Rp. %.1f |\n", h.gaji\_lembur)  WRITE("\t| |")  WRITE("\n\t| Total Gaji \t= Rp. %.1f + Rp. %.1f |\n", h.gaji\_utama, h.gaji\_lembur)  WRITE("\t| \t\t= Rp. %.1f |\n", h.gaji\_total)  WRITE("\t=====================================================\n")  WRITE("\t|| GAJI ANDA \t: Rp. %.1f ||\n", h.gaji\_total)  WRITE("\t=====================================================\n")  CALL ulang()  END  FUNCTION main() : int  DEKLARASI  BEGIN :  WRITE("\t=======================================================\n")  WRITE("\t||\t\tPROGRAM MENGHITUNG GAJI ||\n")  WRITE("\t=======================================================\n")  WRITE("\t|| Ida Bagus Paalakaa RNB""\t\t(2205551003) ||")  WRITE("\n\t|| I Gede Made Rapriananta Pande""\t(2205551005) ||")  WRITE("\n\t|| Ni Kadek Ari Diah Lestari""\t\t(2205551069) ||")  WRITE("\n\t|| I Gusti Ayu Krisna Kusuma Dewi""\t(2205551072) ||")  WRITE("\n\t|| Kadek Yogi Dwi Putra Utama""\t\t(2205551076) ||")  WRITE("\n\t|| Anak Agung Indi Kusuma Putra""\t\t(2205551079) ||")  WRITE("\n\t=======================================================\n")  WRITE("\nTekan apa saja untuk lanjut ")  CALL getch()  system("cls")  CALL perhitungan\_gaji()  return 0  END |

Kode Program 8.1 *Pseudocode* Program Gaji Harian

Kode Program 8.1 adalah *pseudocode* dari menghitung gaji. Perhitungan gaji dimulai dengan mendeklarasikan beberapa variabel menggunakan struct dan mendeklarasikan struct hitung h. Masuk ke fungsi main() kemudian lanjut ke menu untuk meng-*input* jam kerja, kemudian ke menu pilihan untuk menanyakan apakah angka yang diinput sudah benar atau belum. *Input* jam kerja yang kurang dari jam kerja maksimal maka program akan langsung menuju ke fungsi menghitung gaji utama, sedangkan *input* jam kerja yang lebih dari jam kerja maksimal maka akan menuju ke fungsi menghitung gaji lembur. Setelah menuju ke fungsi yang ditentukan maka akan langsung mengeksekusi rumus pada masing-masing fungsi lalu meng-*output-*kan hasil dari perhitungan rumus.

#### Trace Program Gaji Harian

Setelah selesai menyusun alur dan notasi algoritma, *trace* atau pelacakan algoritma pun dapat dilakukan. *Trace* dilakukan untuk melacak alur mulai dari pemberian *input*, memproses *input,* hingga *output* berupa hasil dari proses tersebut, sehingga dapat memastikan apakah implementasi algoritma yang digunakan sudah sesuai atau belum. Berikut ini adalah *trace* dari program gaji.

Tabel 8.1 *Trace* Program Gaji Harian

|  |  |
| --- | --- |
| *Input* | Jam kerja = 5  jam sudah benar = 1  jam kerja = 10  jam sudah benar = 1  mengulang = 2 |
| Proses | if (h.jam\_kerja<=jam\_bekerja){  CALL perhitungan\_gajiUtama()  CALL perhitungan\_gajiUtama()  h.gaji\_utama = h.jam\_kerja \* gaji\_perjam  = 5 \* 10625 = 53125  CALL ulang()  if(h.jam\_kerja>jam\_bekerja){  CALL perhitungan\_gajiLembur()  CALL perhitungan\_gajiLembur()  h.jam\_lembur = h.jam\_kerja - jam\_bekerja;  = 10 – 8 = 2  h.gaji\_lembur = h.jam\_lembur \* jam\_bekerja \* gaji\_perjam;  = 2 \* 8 \* 10625 = 170000  h.gaji\_utama = jam\_bekerja \* gaji\_perjam;  = 8 \* 10625 = 85000  h.gaji\_total = h.gaji\_lembur + h.gaji\_utama;  = 85000 + 170000 = 255000  CALL ulang() |
| *Output* | Input jam kerja:  Apakah jam sudah benar?  Gaji anda : Rp. 53125  Gaji anda : Rp. 255000  Mengulang?  Terima Kasih |

Tabel 8.1 di atas adalah *trace* dari program *gaji* dengan 2 pilihan yaitu jika input jam kerja melebihi maksimal jam bekerja dan kurang dari maksimal jam bekerja. Jam yang kurang akan dihitung sebagai gaji biasa (bukan lembur) dan untuk jam yang lebih akan dihitung sebagai gaji lembur.

### Flowchart, Pseudocode, dan Trace Program Kubus

Program kubus merupakan program yang dapat menghitung luas permukaan dan volume kubus, yang mana ada dua kubus berupa kubus dengan *struct* serta kubus dengan *union*. Masing-masing kubus akan memiliki *field* berupa sisi, luas permukaan, dan volume. Program ini akan menggunakan dua buah variabel saja dalam operasi aritmetikanya, yaitu kubus *union* dan kubus *struct* serta menampilkan *output*-nya. Berikut adalah *flowchart, pseudocode,* serta *trace* dari program kubus.

#### Flowchart Program Kubus

*Flowchart* adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program. Berikut adalah *flowchart* program kubus *struct* dan *union*.

|  |
| --- |
|  |

Gambar 8.4 *Flowchart* Subproses *Cover*

Gambar 8.4 merupakan *flowchart* sub-program dari bagian *cover()* program kubus *struct* dan *union*. Setelah dimulai nama kelompok, nama program, dan tampilan menu akan ditampilkan dan selesai.

|  |
| --- |
|  |

Gambar 8.5 *Flowchart* Subproses *Input*

Gambar 8.5 merupakan *flowchart* sub-program dari *int input()* pada program kubus *struct* dan *union*. Setelah dimulai maka akan dilakukan pengecekan kondisi pada *input* dan jika yang diinputkan adalah bukan bilangan maka kondisi *true* akan dijalankan. Jika yang diinputkan adalah bilangan maka kondisi *false* akan dijalankan dan program akan berakhir.

|  |
| --- |
|  |

Gambar 8.6 *Flowchart* Subproses *Input* Sisi

Gambar 8.6 di atas merupakan *flowchart* sub-program dari *int inputsisi()* pada program kubus *struct* dan *union*. Setelah dimulai maka akan dilakukan pengecekan kondisi pada *input* dan jika yang diinputkan adalah bukan bilangan maka kondisi *true* akan dijalankan. Jika yang diinputkan adalah bilangan maka kondisi *false* akan dijalankan dan program akan berakhir.

|  |
| --- |
|  |

Gambar 8.7 *Flowchart* Subproses *Repeat*

Gambar 8.7 merupakan *flowchart void repeat()* pada program kubus *struct* dan *union*. Setelah dimulai akan ditampilkan pertanyaan terkait pengulangan program, kemudian *user* menginputkan pilihannya dan akan dilakukan pengecekan kondisi. Jika inputnya *‘yes’* maka kondisi true akan dijalankan dan program selesai. Jika inputnya *‘no’* maka kondisi *true* yang selanjutnya menampilkan *“terimakasih”* akan dijalankan dan program selesai. Jika kedua kondisi tidak terpenuhi maka program akan diulang kembali.

|  |
| --- |
|  |

Gambar 8.8 *Flowchart* Subproses Kubus

Gambar 8.8 tersebut merupakan *flowchart void kubus()* pada program kubus *struct* dan *union*. Setelah dimulai *user* akan menginputkan sisi dan kemudian proses perhitungan akan dijalankan dari kubus *struct* kemudian *union*. Setelah selesai, selanjutnya ditampilkan panjang sisi, luas permukaan, volume dan ukuran *memory* dari kubus *struct* dan *union*. Kemudian dilanjutkan dengan dipanggilnya prosedur *repeat* dan program selesai.

|  |
| --- |
|  |

Gambar 8.9 *Flowchart* *Int Main* Kubus *Struct*

Gambar 8.9 di atas merupakan *flowchart main()* pada program kubus *struct* dan *union*. Setelah dimulai akan dilakukan deklarasi *struct* dan *union* beserta variabelnya, kemudian *user* menginputkan pilihan dan akan dipanggil prosedur *input* untuk validasi. *Input* ini akan dicek melalui pengecekan kondisi, jika *false* maka akan diminta pengulangan *input*. Sebaliknya jika *true* maka akan dilakukan pengecekan kondisi *input* selanjutnya terkait pilihan, jika pilihan *‘1’* maka akan dilanjutkan dengan pemanggilan prosedur kubus dan program selesai. Jika pilihan *‘2’* maka akan ditampilkan *‘terimakasih’* dan program akan selesai. Jika kedua kondisi tidak terpenuhi maka program akan diulang.

#### Pseudocode Program Kubus

*Pseudocode* adalah suatu rangkaian pernyataan atau deklarasi yang ditulis dalam bahasa pemrograman komputer yang terbaca manusia. Kode sumber yang menyusun suatu program biasanya disimpan dalam satu atau lebih berkas teks, dan dapat pula ditampilkan dalam bentuk cuplikan kode (*code snippet*) yang dicetak. Berikut *pseudocode* program menghitung luas permukaan dan volume kubus.

|  |
| --- |
| Program Menghitung Luas Permukaan dan Volume Kubus dengan Struct dan Union    DEKLARASI  Panjangsisi : INTEGER  Pilihan : INTEGER    typedef struct {  panjang, luasPermukaan, volume : INTEGER  }kubusStruct    typedef union{  panjang, luasPermukaan, volume : INTEGER  }kubusUnion    FUNCTION input : int  DEKLARASI  cek : char    IF(cek=scanf("%d%c", &pilihan, &cek)!=2 || pilihan<1) THEN  System(“cls”)  fflush(stdin)  WRITE("\n\t\tMASUKKAN ANGKA YANG BENAR !")  WRITE("\n\t\tMasukkan pilihan : ")  return input()  ELSE  return panjangsisi  END IF  END    FUNCTION inputsisi() :int  DEKLARASI  Cek : char    IF(cek=scanf("%d%c", &panjangsisi, &cek)!=2 || panjangsisi<0) THEN  system("cls")  fflush(stdin)  WRITE("\n\t\tMASUKKAN ANGKA YANG BENAR !")  WRITE("\n\t\tMasukkan sisi : ")  return inputsisi()  ELSE  return panjangsisi  END IF  END    FUNCTION repeat() :void  DEKLARASI  pil[3] : char  WRITE("\nApakah anda ingin mengulang program ini? (yes/no) \n")  READ(pil)    IF(strcmp(pil,"yes")==0) THEN  system("cls")  CALL main()  ELSE IF(strcmp(pil,"no")==0) THEN  system("cls")  WRITE("\n")  write("+===========================================+\n");  write("| Terima Kasih |\n");  write("+===========================================+\n\n");  write("|===========================================|\n");  write("| KELOMPOK 23 |\n");  write("|===========================================|\n");  write("| Nama Anggota Kelompok |\n");  write("|===========================================|\n");  write("|2205551005 Gede Made Rapriananta Pande |\n");  write("|2205551003 Ida Bagus Paalakaa RNB |\n");  write("|2205551069 Ni Kadek Ari Diah Lestari |\n");  write("|2205551072 I Gusti Ayu Krisna Kusuma Dewi |\n");  write("|2205551076 Kadek Yogi Dwi Putra Utama |\n");  write("|2205551079 Anak Agung Indi Kusuma Putra |\n");  write("|===========================================|\n\n");  CALL exit(1)  ELSE  WRITE("Silahkan Masukkan Pilihan Yang Benar!! ")  CALL repeat();  END IF  CALL getch()  END    FUNCTION kubus() :void  WRITE("\t\tMasukkan sisi : ")  CALL inputsisi()  WRITE("\n")  kubusStruct structs = {.panjang <- panjangsisi,  .luasPermukaan <- 6 \* panjangsisi \* panjangsisi,  .volume <- panjangsisi \* panjangsisi \* panjangsisi}    kubusUnion unions[3]  unions[0].panjang <- panjangsisi  unions[1].luasPermukaan <- 6 \* panjangsisi \* panjangsisi  unions[2].volume <- panjangsisi \* panjangsisi \* panjangsisi  WRITE("\t\tPanjang Sisi : %d\n", structs.panjang)  WRITE("\t\tLuas Permukaan : %d\n", structs.luasPermukaan)  WRITE("\t\tVolume : %d\n\n", structs.volume)  WRITE("\t\tPanjang Sisi : %d\n", unions[0])  WRITE("\t\tLuas Permukaan : %d\n", unions[1])  WRITE("\t\tVolume Kubus : %d\n\n", unions[2])  WRITE("\t\tUkuran Memory pada Kubus Struct : %d\n\n", sizeof(structs))  WRITE("\t\tUkuran Memory pada Kubus Union : %d\n\n", sizeof(unions[0]))  CALL repeat()  END    FUNCTION cover() :void  write("+===========================+\n");  write("| PROGRAM PERHITUNGAN KUBUS |\n");  write("| DENGAN STRUCT & UNION |\n");  write("+===========================+\n\n");  write("+===========================+\n");  write("| MAIN MENU |\n");  write("+===========================+\n");  write("| 1. Kubus Struct dan Union |\n");  write("| 2. Tutup program |\n");  write("+===========================+\n\n");  END    FUNCTION main() : int  system("color B")  CALL cover();    WRITE("\n\t\tMasukkan pilihan : ")  CALL input()      IF(pilihan == 1) THEN  system("cls")  CALL kubus()  ELSE IF(pilihan == 2) THEN  WRITE("\n");  write("+===========================================+\n");  write("| Terima Kasih |\n");  write("+===========================================+\n\n");  write("|===========================================|\n");  write("| KELOMPOK 23 |\n");  write("|===========================================|\n");  write("| Nama Anggota Kelompok |\n");  write("|===========================================|\n");  write("|2205551005 Gede Made Rapriananta Pande |\n");  write("|2205551003 Ida Bagus Paalakaa RNB |\n");  write("|2205551069 Ni Kadek Ari Diah Lestari |\n");  write("|2205551072 I Gusti Ayu Krisna Kusuma Dewi |\n");  write("|2205551076 Kadek Yogi Dwi Putra Utama |\n");  write("|2205551079 Anak Agung Indi Kusuma Putra |\n");  write("|===========================================|\n\n");  CALL exit(1)  ELSE  system("cls")  WRITE("\t\tMASUKKAN ANGKA YANG TEPAT !\n")  CALL main()  END IF  return 0  END |

Kode Program 8.2 *Pseudocode* Program Kubus *Struct* dan *Union*

Kode program 8.2 merupakan *pseudocode* dari program kubus *struct* dan *union* dengan menggunakan bahasa C. Di mana pada kode program di atas menjelaskan proses jalannya program menghitung kubus.

#### Trace Program Kubus

Proses pembuatan program kubus *struct* dan *union* terdiri dari beberapa tahapan. Tahapan-tahapan tersebut antara lain pembuatan pembuatan kode program, dan melakukan *trace* pada program yang telah dibuat. Tahapan-tahapan tersebut dapat dijelaskan sebagai berikut.

Tabel 8.2 *Trace* Program Kubus *Struct* dan *Union*

|  |  |
| --- | --- |
| *Input* | pilihan = 1  panjangsisi = 10  pil = no |
| Proses | kubusStruct struct = .panjang = 10  .luasPermukaan = 6\*10\*10  .volume = 10\*10\*10  kubusUnion union[3]  unions[0].panjang = 10  unions[1].luasPermukaan = 6\*10\*10  unions[2].volume = 10\*10\*10  sizeof(structs)  sizeof(unions) |
| *Output* | Print :  Panjang Sisi : 10  Luas Permukaan : 600  Volume : 1000    Panjang Sisi : 10  Luas Permukaan : 600  Volume : 1000    Ukuran Memory pada Kubus Struct : 12  Ukuran Memory pada Kubus Union : 4 |

Tabel 8.2 merupakan *trace* dari program kubus *struct* dan *union*, pada *input* *user* memasukkan pilihan 1 untuk melakukan perhitungan dan dilanjutkan dengan *input* panjang sisi dan pilihan no agar program tidak diulang. Kemudian pada proses dilakukan perhitungan luas permukaan dan volume serta menghitung ukuran memori dari kubus *struct* dan union. Pada *output* akan ditampilkan panjang sisi, luas permukaan, volume dan ukuran memori pada kedua kubus.

### Flowchart, Pseudocode, dan Trace Program Palindrom

Program palindrom merupakan sebuah program yang dapat menerima *input* kata yang kemudian akan menentukan banyaknya huruf dalam kata tersebut serta menentukan apakah kata tersebut merupakan palindrom atau bukan. Hasil dari proses tersebut nantinya akan tersimpan pada sebuah *file txt* (kamuskata.txt) yang isinya tidak akan hilang bila program ditutup dan dapat ditambahkan dengan data baru di kemudian hari. Berikut adalah *flowchart, pseudocode,* serta *trace* dari program palindrom.

#### Flowchart Program Palindrom

*Flowchart* program *palindrom* memiliki fungsi untuk memvisualisasikan alur alur dari program Enkripsi dan Dekripsi. Hal ini berguna untuk memudahkan *programmer* untuk membuat program karena algoritma dari program sudah divisualisasikan sehingga akan mudah dipahami oleh *programmer*. Alur *flowchart* *palindrom* bisa dilihat pada gambar.

|  |
| --- |
|  |

Gambar 8.10 *Flowchart Int Main* Program *Palindrom*

Gambar 8.10 di atas merupakan *flowchart* dari fungsi *main* pada program *palindrom*. Alur dari *flowchart* fungsi main program *palindrom* dimulai dari *“mulai”.* Lalu akan menampilkan nama kelompok dan lanjut ke prosedur *main menu*.

|  |
| --- |
|  |

Gambar 8.11 *Flowchart Main Menu*

Gambar 8.11 di atas merupakan *flowchart* dari prosedur *mainMenu()* program *palindrom* di mana pengguna akan diminta untuk menginputkan. Kata yang akan dicek apakah bersifat *palindrom* atau tidak. Setelah selesai akan masuk prosedur ulang.

|  |
| --- |
|  |

Gambar 8.12 *Flowchart Palindrom*

Gambar 8.12 di atas merupakan *flowchart* dari prosedur *palindrom()* yang digunakan untuk menjalankan proses untuk mengecek apakah kata yang diinput bersifat *palindrom* atau tidak. Fungsi ini akan mengembalikan nilai *true* apabila kata bersifat *palindrom* dan mengembalikan nilai *false* jika kata tidak bersifat *palindrom*.

|  |
| --- |
|  |

Gambar 8.13 *Flowchart Frequency*

Gambar 8.13 di atas merupakan *flowchart* dari prosedur *frequency()* yang digunakan untuk menghitung frekuensi munculnya huruf pada kata yang diinputkan, hasilnya akan ditampilkan sesuai dengan urutan huruf pada kata yang diinputkan.

|  |
| --- |
|  |

Gambar 8.14 *Flowchart Display*

Gambar 8.14 di atas merupakan *flowchart* dari prosedur *display().* Yang digunakan untuk menampilkan hasil dari proses yang dilakukan oleh fungsi sebelumnya. Prosedur ini akan menampilkan kata yang diinputkan, panjang kata yang diinputkan, frekuensi munculnya kata pada kata yang diinputkan dan apakah kata yang diinputkan bersifat *palindrom* atau bukan.

|  |
| --- |
|  |

Gambar 8.15 *Flowchart* Ulang

Gambar 8.15 di atas merupakan *flowchart* dari fungsi *ulang()* yang digunakan untuk menampilkan menu ulang yang akan memberikan pilihan kepada pengguna apakah ingin mengulang program atau tidak. Jika pengguna tidak ingin mengulang maka menu terima kasih akan ditampilkan dan program akan selesai.

|  |
| --- |
|  |

Gambar 8.16 *Flowchart* Validasi *Input*

Gambar 3.89 di atas merupakan *flowchart* dari validasi yang digunakan pada program konversi biner. Fungsi *validasiRange()* digunakan untuk memvalidasi bilangan dengan *range* tertentu. Fungsi *validasiKey()* digunakan untuk memvalidasi bilangan positif.

#### Pseudocode Program Palindrom

*Pseudocode* program *palindrom* berisi perintah-perintah yang akan digunakan untuk membuat program *palindrom*. *Pseudocode* tidak terpaku pada peraturan dari Bahasa pemrograman mana pun, jadi pada dasarnya *pseudocode* merupakan orek-orek awal bagi *programmer* untuk membuat program yang diinginkan.

|  |
| --- |
| // Program Palindrom  hasil : bool  FUNCTION main() : int  DESKRIPSI:  BEGIN  CALL namaKelompok()  CALL mainMenu()  END  FUNCTION palindrom(char string[]) : bool  DEKLARASI:  len, middle, salah : int  DESKRIPSI:  BEGIN  len = strlen(string)-1  middle = len / 2  FOR(int i = 0; i < strlen(string);i++)  IF (i == middle-1) THEN  BREAK  ELSE IF (string[i] != string[len-1]) THEN  salah++  len--  ENDFOR  IF( salah > 0) THEN  return false  ELSE  return true  END  FUNCTION frequency(char \*string, int frequency[]) : int  DEKLARASI:  c : char  len, whitespace : int  DESKRIPSI:  BEGIN  len = strlen(string)-1  IF (len < 2) THEN  CALL pesanError()  FOR (int i = 0; i < len; i++)  IF ((int)string[i] == 32 || (int)string[i] == 10 || (int)string[i] == 9 ) THEN  whitespace++  c = tolower(string[i])  IF( (int)c >= 97 && (int)c <= 122) THEN  frequency[c-97]++  ELSE  CALL pesanError()  ENDFOR  IF (whitespace > 0)  CALL pesanError()    END  FUNCTION display(char \*string, int frequency[]) : int  DEKLARASI:  redundansi, len : int  a : char  DESKRIPSI:  BEGIN  len = strlen(string)-1;  FILE \*f = fopen("kamuskata.txt","ab")  WRITE "$string($len) : "  FOR (int i = 0; i < len; i++)  a[i] = string[i]  redundansi = 0  FOR (int j =0;j<i;j++)  IF (a[i] == string[j]) THEN  redundansi++  ENDFOR  IF (redundansi == 0) THEN  WRITE "$string[i] = $frequency[string[i]-97]"  ENDFOR  IF (hasil == true)  WRITE ": Palindrom\n"  ELSE  WRITE ": bukan Palindrom"  fclose(f)  END  PROCEDURE pesanError() : void  DESKRIPSI:  BEGIN  CALL system("cls")  WRITE " Maaf Input Anda Salah !"  WRITE " Mohon hanya inputkan huruf "  CALL system("pause")  CALL mainMenu()  END  PROCEDURE mainMenu() : void  DEKLARASI:  pilihan, read : int  string : char  DESKRIPSI:  BEGIN  DO  letterFrequency : int  CALL system("cls");  WRITE " PALINDROM CHECKER"  WRITE " Masukan kata : "  READ string  hasil = palindrom(string)  CALL frequency(string,letterFrequency)  CALL display(string,letterFrequency)  CALL system("cls")  WRITE " PROGRAM SUKSES!!!"  WRITE "untuk hasil bisa di cek pada kamuskata.txt"  CALL system("pause")  read = ulang()  WHILE(read == 1)  END  PROCEDURE namaKelompok() : void  DESKRIPSI:  BEGIN  CALL system("cls");  WRITE "namaKelompok"  CALL system("pause")  CALL system("cls")  END  FUNCTION ulang() : int  DEKLARASI:  pilihan : int  DESKRIPSI:  BEGIN  CALL system("cls")  WRITE " Apakah anda ingin mencoba lagi?"  WRITE " (1 = ya / 2 = tidak)"  pilihan = validasiRange()  SWITCH (pilihan)  CASE 1:  CALL system("cls")  return 1  CASE 2:  CALL goodBye()  DEFAULT:  WRITE " Maaf Input anda salah !"  WRITE " Masukan Input Dengan Benar"  CALL ulang()  ENDSWITCH  END  FUNCTION validasiRange() : int  DEKLARASI:  input, notValid : char  valid : int  READ input, notValid  DESKRIPSI:  BEGIN  IF (sscanf(input, "%d%c", &valid, &notValid) == 1 && valid == 1 || valid == 2) THEN  return valid  ELSE  CALL system("cls");  WRITE " Maaf Input anda salah !"  WRITE " Masukan Input Dengan Benar"  CALL validasiRange()  END  FUNCTION goodBye() : void  DESKRIPSI:  BEGIN  CALL system("cls")  WRITE "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n"  WRITE "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n"  WRITE "\t\t| | | |\n"  WRITE "\t\t| | | |\n"  WRITE "\t\t| | Terima Kasih Telah Menggunakan Program Ini yaaaaaa!!! | |\n"  WRITE "\t\t| | | |\n"  WRITE "\t\t| | | |\n"  WRITE "\t\t| | \\|||||||||||||||||||| |||||||||||||||||\\ | |\n"  WRITE "\t\t| | \\||||||||||||||||||| ||||||||||||||||||\\ | |\n"  WRITE "\t\t| | \\|||||||||||||||||| |||||||||||||||||||\\ | |\n"  WRITE "\t\t| | |||||||||| | |\n"  WRITE "\t\t| | |||||||||| |||||||||||||||||||/ | |\n"  WRITE "\t\t| | |||||||||| ||||||||||||||||||/ | |\n"  WRITE "\t\t| | |||||||||| |||||||||||||||||/ | |\n"  WRITE "\t\t| | |||||||||| ||||||||| | |\n"  WRITE "\t\t| | |||||||||| ||||||||| | |\n"  WRITE "\t\t| | |||||||||| ||||||||| | |\n"  WRITE "\t\t| | |||||||||| ||||||||| | |\n"  WRITE "\t\t| | \\|||||||| |||||||/ | |\n"  WRITE "\t\t| | \\||||||| ||||||/ | |\n"  WRITE "\t\t| | \\|||||| |||||/ | |\n"  WRITE "\t\t| | \\||||| ||||/ | |\n"  WRITE "\t\t| | \\|||| |||/ | |\n"  WRITE "\t\t| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| |\n"  WRITE "\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n"  END |

Kode Program 8.3 *Pseudocode* Program *Palindrom*

Kode Program 8.3 di atas merupakan *pseudocode* dari program *palindrom*. Pengguna akan menginputkan kata yang akan dicek apakah bersifat *palindrom* atau bukan. Setelah program akan berjalan dan kata yang dinputkan, panjang kata, frekuensi munculnya huruf pada kata dan apakah kata yang diinputkan itu bersifat *palindrom* atau bukan akan disimpan sebagai *output* pada *file kamuskata.txt.*

#### Trace Program Palindrom

*Trace* pada program merupakan salah satu hal yang penting. *Trace* berguna untuk mengetahui alur program bekerja dan dapat digunakan untuk mendeteksi kesalahan yang ada pada program sehingga kesalahan bisa ditemukan dan diperbaiki. *Trace* pada program *palindrom* dapat dilihat dibawah ini.

Tabel 8.3 *Trace* Program Palindrom

|  |  |
| --- | --- |
| *Input* | Masukan kata = malam  Ingin Mencoba Menginput Kembali (1/2) = 2 |
| Proses | CALL palindrom()  for(int i = 0; i < strlen(string);i++){  if( i == middle-1) break;  else if(string[i] != string[len-1]) salah++;  len--;  }  if( salah > 0) return false;  else return true;  CALL frequency()  if(len < 2) pesanError();  for(int i = 0; i < len; i++){  if((int)string[i] == 32 || (int)string[i] == 10 || (int)string[i] == 9 )whitespace++;  c = tolower(string[i]);  if( (int)c >= 97 && (int)c <= 122) frequency[c-97]++;  else pesanError();  }  if(whitespace > 0) pesanError();  CALL display()  FILE \*f = fopen("kamuskata.txt","ab");  fprintf(f,"%s(%d) : ",string,len);  for(int i = 0; i < len; i++){  a[i] = string[i];  redundansi = 0;  for(int j =0;j<i;j++){  if(a[i] == string[j]) redundansi++;  }  if(redundansi == 0) fprintf(f,"%c = %d ",string[i],frequency[string[i]-97]);  }  if (hasil == true) fprintf(f,": Palindrom\n");  else fprintf(f,": bukan Palindrom\n");  fclose(f); |
| *Output* | Tampilkan program sukses  Terima kasih |

Tabel 8.3 di atas menjelaskan bagaimana program ini berjalan, pertama pengguna akan meng-*input* kata yang akan dicek oleh program apakah bersifat *palindrom* atau tidak, lalu program akan mengecek apakah kata tersebut bersifat palindrom atau bukan, kata yang di-*input*-kan pada *trace* ini adalah “malam”. Program akan mengecek apakah kata tersebut bersifat *palindrom*, mengecek jumlah frekuensi munculnya karakter pada kata tersebut. Dan menulis hasilnya pada *file kamuskata.txt*. Lalu program akan menampilkan menu ulang untuk menanyakan pengguna apakah ingin mengulang program (1 = ya, 2 = tidak), jika pengguna memilih 1 maka program akan kembali berulang, namun jika pengguna memilih 2 maka menu terima kasih akan muncul dan program akan berakhir.

### Flowchart, Pseudocode, dan Trace Program Data Mahasiswa

Program data mahasiswa merupakan suatu program sistem informasi data mahasiswa yang berisikan NIM serta nama mahasiswa. Program akan menyimpan data yang diberikan di dalam sebuah *file* (*file txt*) yang mana *file* tersebut dapat ditemukan pada *drive* mana pun. Beberapa ketentuan lain seperti data tidak hilang, dapat melakukan pencarian data, serta dapat di-*update* dan mencegah duplikasi data pun dapat dilakukan oleh program ini. Berikut adalah *flowchart, pseudocode,* serta *trace* dari program data mahasiswa.

#### Flowchart Program Data Mahasiswa

*Flowchart* adalah sekumpulan simbol-simbol yang menjelaskan mengenai suatu alur proses atau langkah-langkah yang berurutan dari program tersebut yang harus diikuti oleh pemrogram. Dapat disimpulkan, bahwa *flowchart* digunakan untuk menunjukkan alur proses dari sebuah program secara sistematis hingga selesai.

|  |
| --- |
|  |

Gambar 8.17 *Flowchart* Subproses Keluar

Gambar 8.17 di atas adalah *flowchart* dari subproses *keluar()*. Di mana dalam *flowchart* tersebut terdapat sistem untuk membersihkan layar, *output* untuk mengatakan terima kasih, dan sistem untuk keluar.

|  |
| --- |
|  |

Gambar 8.18 *Flowchart* Subproses Menu Utama

Gambar 8.18 diatas adalah *flowchart* subproses *menuUtama()*. *Flowchart* dimulai dengan pendeklarasian variabel pilihan, kemudian menampilkan pilihan menu, dan melakukan masukkan *input*. Kemudian terdapat perkondisian berdasarkan pilihan, dan setiap pilihan masuk ke dalam subproses, dan *flowchart* selesai.

|  |
| --- |
|  |

Gambar 8.19 *Flowchart Int Main*

Gambar 8.19 di atas adalah *flowchart main* dari program data mahasiswa. Dalam *flowchart* terdapat penampilan nama kelompok dan subproses *menuUtama()*, kemudian terdapat *return 0*, dan *flowchart* selesai.

|  |
| --- |
|  |

Gambar 8.20 *Flowchart* Subproses Pendeklarasian *Struct*

Gambar 8.20 di atas adalah *flowchart* subproses pendeklarasian *struct*. Di mana dalam *flowchart* tersebut, *struct* dideklarasikan dengan nama *mahasiswa*, kemudian *struct* tersebut terdiri dari variabel *nim* dan *nama* dengan tipe data yang berbeda.

|  |
| --- |
|  |

Gambar 8.21 *Flowchart* Subproses *Input*

Gambar 8.21 di atas adalah *flowchart* subproses *input()*. Dalam *flowchart* dimulai dengan pendeklarasian berbagai variabel. Kemudian terdapat penginputan variabel *masukkan*, dan dalam *flowchart* tersebut terdapat juga perulangan dan perkondisian.

|  |
| --- |
|  |

Gambar 8.22 *Flowchart* SubprosesPilih Ulang

Gambar 8.22 diatas adalah *flowchart* subproses *pilihUlang()*. Di mana *flowchart* dimulai dengan pendeklarasian variabel *hurufUlang*, kemudian terdapat pertanyaan apakah ingin mengulang, dan *user* diminta untuk memasukkan inputan, berupa *y* untuk mengulang, *b* untuk kembali ke menu, dan *o* untuk keluar. Dalam *flowchart* tersebut juga terdapat validasi dari ketiga masukkan tersebut.

|  |
| --- |
|  |

Gambar 8.23 *Flowchart* Subproses Pertanyaan Hapus

Gambar 8.23 diatas adalah *flowchart* subproses *pertanyaanHapus()*. *Flowchart* dimulai dengan pendeklarasian variabel *hurufHapus dan hasil*, kemudian terdapat pertanyaan apakah ingin menghapus, dan user diminta memasukkan inputan, berupa y untuk iya, dan n untuk tidak. Dalam *flowchart* juga terdapat validasi dari masukkan tersebut.

|  |
| --- |
|  |

Gambar 8.24 *Flowchart* Subproses *SetDefaultDataTmp*

Gambar 8.24 diatas adalah *flowchart* subproses *setDefaultDataTmp()*. Setelah dimulai maka akan dilakukan pendeklarasian variabel *i*, kemudian terdapat perulangan *for*, jika kondisi pada perulangan bernilai *true*, maka proses di dalam perulangan tersebut akan dieksekusi, namun jika bernilai *false* *flowchart* berakhir.

|  |
| --- |
|  |

Gambar 8.25 *Flowchart* SubprosesValidasi NIM

Gambar 8.25 di atas adalah *flowchart* subproses *validasiNim()*. *Flowchart* dimulai dengan pendeklarasian beberapa variabel, kemudian *user* diminta memasukkan *input* berupa angka, di mana angka hanya bisa dari nomor 0-9, dan jumlahnya sebanyak 10 angka, jika di luar itu maka akan diminta meng-*input* ulang. Jika proses berhasil maka *flowchart* selesai.

|  |
| --- |
|  |

Gambar 8.26 *Flowchart* Subproses Validasi Nama

Gambar 8.26 diatas adalah *flowchart* subproses *validasiNama()*. *Flowchart* dimulai dengan pendeklarasian variabel *i* dan *salah*. Kemudian terdapat perulangan untuk variabel *masukkan*, dan kondisi dari variabel tersebut hanya bisa untuk inputan huruf, spasi, koma, dan titik. Setelah proses tersebut selesai, maka *flowchart* selesai.

|  |
| --- |
|  |

Gambar 8.27 *Flowchart* Subproses Cek NIM

Gambar 8.27 di atas menggambarkan *flowchart* subproses *cekNim()*. Setelah dimulai maka akan dilakukan pendeklarasian variabel *i*, dan *ditemukan*, kemudian terdapat perulangan dan perkondisian, di mana perkondisian ini digunakan untuk mengecek apakah nim tersebut sudah ada atau belum. Setelah proses tersebut selesai, sebelum *flowchart* berakhir terdapat pengembalian variabel *ditemukan*.

|  |
| --- |
|  |

Gambar 8.28 *Flowchart* Subproses Cek NIM Edit

Gambar 8.28 diatas merupakan *flowchart* subproses *cekNimEdit()*. Setelah dimulai maka akan dilakukan pendeklarasian variabel *i*, dan *ditemukan*, kemudian terdapat perulangan dan perkondisian. Setelah proses tersebut selesai, sebelum *flowchart* berakhir terdapat pengembalian variabel *ditemukan*.

|  |
| --- |
|  |

Gambar 8.29 *Flowchart* Subproses *ProsesReadDataTmp*

Gambar 8.29 diatas merupakan *flowchart* subproses *prosesReadDataTmp()*. *Flowchart* dimulai dengan mendeklarasikan variabel *jumlah*. Kemudian terdapat fungsi *file* *fopen* yang memiliki fungsi untuk membuka *file*, kemudian terdapat perkondisian dan perulangan. Di dalam perulangan tersebut terdapat fungsi *file* *feof* yang berfungsi mendeteksi akhir *file*, kemudian terdapat juga fungsi *file* *fscanf* untuk membaca *string* maupun bilangan, dan jika proses tersebut sudah selesai, maka *file* ditutup dengan fungsi *file* *fclose*. Sebelum *flowchart* berakhir, maka akan dilakukan pengembalian variabel *jumlah* terlebih dahulu.

|  |
| --- |
|  |

Gambar 8.30 *Flowchart* Subproses Proses Tambah Data

Gambar 8.30 diatas merupakan *flowchart* subproses *prosesTambahData()*. Setelah dimulai maka akan dilakukan pembukaan *file* *dataMahasiswa*, kemudian penulisan *file*, dan terakhir dilakukan penutupan *file*.

|  |
| --- |
|  |

Gambar 8.31 *Flowchart* Subproses Proses Lihat Data

Gambar 8.31 diatas merupakan *flowchart* subproses *prosesLihatData()*. Setelah dimulai maka akan dilakukan pendeklarasian variabel *i*, kemudian terdapat perkondisian yang menyatakan tidak ada data mahasiswa, jika kondisi tersebut bernilai *false* maka lanjut ke perulangan untuk menampilkan data, jika semua data sudah ditampilkan, maka *flowchart* berakhir.

|  |
| --- |
|  |

Gambar 8.32 *Flowchart* Subproses Proses Perbaharui Data

Gambar 8.32 di atas merupakan *flowchart* subproses *prosesPerbaharuiData()*. *Flowchart* dimulai dengan pendeklarasian variabel *i*, dan *j*, kemudian dilanjutkan ke dalam prosesnya yaitu menuliskan data yang baru.

|  |
| --- |
|  |

Gambar 8.33 *Flowchart* Subproses Proses Hapus Data

Gambar 8.33 di atas merupakan *flowchart* subproses *prosesHapusData()*. *Flowchart* dimulai dengan pendeklarasian variabel *i*, kemudian dilanjutkan ke dalam prosesnya yaitu penghapusan data yang sudah ada pada *file* sebelumnya.

|  |
| --- |
|  |

Gambar 8.34 *Flowchart* Subproses Tambah Data

Gambar 8.34 di atas merupakan *flowchart* subproses *tambahData()*. *Flowchart* dimulai dengan mendeklarasikan beberapa variabel dan *struct*. Kemudian *user* diminta untuk memasukkan NIM, di mana pada proses ini terdapat validasi dan juga perkondisian serta perulangan di mana nim yang sama tidak bisa lagi di-*input*-kan, sehingga *user* akan meng-*input* kembali. Selanjutnya *user* diminta untuk memasukkan nama, dalam proses ini juga terdapat validasi. Jika proses tersebut selesai, maka data berhasil di-*input*-kan, dan *flowchart* selesai.

|  |
| --- |
|  |

Gambar 8.35 *Flowchart* Subproses Lihat Data

Gambar 8.35 di atas merupakan *flowchart* subproses *lihatData()*. *Flowchart* dimulai dengan pendeklarasian *struct*, kemudian *flowchart* dilanjutkan dengan proses menampilkan data mahasiswa, jika proses tersebut sudah selesai, maka *flowchart* selesai.

|  |
| --- |
|  |

Gambar 8.36 *Flowchart* Subproses Perbaharui Data

Gambar 8.36 di atas merupakan *flowchart* subproses *perbaharuiData()*. *Flowchart* diawali dengan pendeklarasian beberapa variabel dan *struct*. Kemudian akan ditampilkan daftar data mahasiswa, selanjutnya *user* diminta memilih data yang ingin diperbaharui, jika sudah memilih masukkan data nim dan nama yang baru, pada proses ini juga terdapat validasi dan jika *true,* maka data berhasil diperbaharui. Jika belum terdapat data maka *user* akan diminta memasukkan data terlebih dahulu. Dan *flowchart* subproses ini selesai.

|  |
| --- |
|  |

Gambar 8.37 *Flowchart* Subproses Hapus Data

Gambar 8.37 di atas merupakan *flowchart* subproses *hapusData()*. Setelah mulai *flowchart* akan mendeklarasikan *struct*, kemudian akan ditampilakan data mahasiswa. *User* diminta untuk memilih data yang ingin dihapus, jika sudah memilih selanjutnya akan ada pertanyaan untuk yakin mengapus data, jika iya maka data berhasil dihapus begitu juga sebaliknya. Jika belum terdapat data maka *user* diminta untuk memasukkan data terlebih dahulu. Dan *flowchart* subproses selesai.

#### Pseudocode Program Data Mahasiswa

*Pseudocode* adalah suatu cara menulis algoritma yang mirip dengan bahasa pemrograman, tetapi *pseudocode* lebih sederhana dan mudah dipahami oleh manusia pada umumnya. *Pseudocode* memudahkan *user* untuk menyelesaikan suatu masalah tanpa memikirkan sintaks dari suatu bahasa pemrograman. Berikut ini *pseudocode* dari program data mahasiswa.

|  |
| --- |
| Algoritma Program Data Mahasiswa  PROTOTYPE FUNCTION:  input(): float  pertanyaanHapus(): int  validasiNim(): int  cekNim(): int  cekNimEdit(): int  prosesReadDataTmp: int  main(): int  keluar(): void  setDefaultDataTmp(): void  prosesTambahData(): void  prosesLihatData(): void  prosesPerbaharuiData(): void  prosesHapusData(): void  tambahData(): void  lihatData(): void  perbaharuiData(): void  hapusData(): void  menuUtama(): void  FUNCTION mahasiswa: struct  DEKLARASI:  nim: int  nama[50]: char  FUNCTION input(int desimal, int min, int batasAkhur): float  DEKLARASI:  masukkan[100]: char  i = 0: int  belakang = 0: int  isiDepan = 0: int  isiBelakang = 0: int  status = 0: int  titik = 0: int  minus = 0: int  salah = 0: int  hasil: float  DESKRIPSI:  BEGIN  read(masukkan)  fflush(stdin)  WHILE(masukkan[i] != '\0') THEN  IF(masukkan[0] == '.') THEN  salah = 1  masukkan[0] = '\0'  ELSE IF(masukkan[i]>='0' && masukkan[i]<='9' || masukkan[i] == '.' || (masukkan[0] == '-' && min == 1)) THEN  IF((masukkan[i] == '.') && (desimal == 1)) THEN  titik = titik + 1  IF(titik > 1 || masukkan[i+1] == '\0') THEN  salah = 1  masukkan[i] = '\0'  END IF  status = 1  ELSE IF((masukkan[i] == '.') && (desimal == 0)) THEN  salah = 1  masukkan[i] = '\0'  END IF    IF(masukkan[0] == '-' && min == 1) THEN  minus = 1  ELSE IF(masukkan[0] == '-' && min == 0) THEN  salah = 1  masukkan[i] = '\0'  END IF  IF(masukkan[i]>='0' && masukkan[i]<='9') THEN  IF(status == 1) THEN  isiBelakang = (isiBelakang \* 10) + (masukkan[i] - 48)  belakang = belakang + 1  ELSE THEN  isiDepan = (isiDepan \* 10) + (masukkan[i] - 48)  END IF  END IF  i++  ELSE THEN  salah = 1  masukkan[i] = '\0'  END IF  END WHILE  IF(titik == 1) THEN  hasil = (float)isiDepan + (isiBelakang / (pow(10, belakang)))  ELSE THEN  hasil = isiDepan  END IF  IF(minus == 1 && min == 1) THEN  hasil = hasil - (hasil \* 2)  ELSE IF(minus == 1 && min == 0) THEN  salah = 1  masukkan[i] = '\0'  END IF  IF(salah == 1) THEN  write "input salah, masukkan angka yang benar: "  RETURN input(desimal, min, batasAkhir)  ELSE IF(min == 0 && hasil == 0) THEN  write "input tidak boleh 0, masukkan angka yang benar: "  RETURN input(desimal, min, batasAkhir)  ELSE IF(batasAkhir != 0 && hasil > batasAkhir) THEN  write "input tidak boleh lebih dari %d, masukkan angka yang benar: ", batasAkhir  ELSE THEN  RETURN hasil  END IF  END  FUNCTION pilihanUlang(int ulang): void  DEKLARASI:  hurufUlang[1]: char  DESKRIPSI:  BEGIN  write "apakah ingin mengulang proses? y/b/o: "  read(hurufUlang)  fflush(stdin)  IF((strcmp(hurufUlang, "y" ) != 0) && (strcmp(hurufUlang, "Y") != 0) && (strcmp(hurufUlang, "b") != 0) && (strcmp(hurufUlang, "B" ) != 0) && (strcmp(hurufUlang, "o" ) != 0) && (strcmp(hurufUlang, "O" ) != 0)) THEN  write "masukkan anda salah, masukkan dengan benar"  pilihanUlang(ulang)  ELSE IF((strcmp(hurufUlang, "y") == 0)) THEN  IF(ulang == 1) THEN  tambahData()  ELSE IF(ulang == 2) THEN  lihatData()  ELSE IF(ulang == 3) THEN  perbaharuiData()  ELSE IF(ulang == 4)  hapusData()  END IF  ELSE IF((strcmp(hurufUlang, "b") == 0)) THEN  menuUtama()  ELSE THEN  keluar()  END IF  END  FUNCTION pertanyaanHapus(): int  DEKLARASI:  hurufHapus[1]: char  hasil = 0: int  DESKRIPSI:  BEGIN  write "apakah yakin menghapus data ini? y/n: "  read(hurufHapus)  fflush(stdin)  IF((strcmp(hurufHapus, "y") != 0) && (strcmp(hurufHapus, "n") != 0)) THEN  write "masukkan anda salah, masukkan dengan benar"  pertanyaanHapus()  ELSE IF((strcmp(hurufHapus, "y") == 0)) THEN  hasil = 1  ELSE THEN  hasil = 0  END IF    RETURN hasil  END  FUNCTION setDefaultDataTmp(struct mahasiswa dataTmp[]): void  DEKLARASI:  i: int  DESKRIPSI:  BEGIN  FOR(i = 0; i < 100; i++) THEN  dataTmp[i].nim = 0  END FOR  END  FUNCTION validasiNim(): int  DEKLARASI:  masukkan[100]: char  i = 0: int  salah = 0: int  hasil = 0: int  DESKRIPSI:  BEGIN  read(masukkan)  fflush(stdin)  WHILE(masukkan[i] != '\0') THEN  IF(masukkan[i] >= '0' && masukkan[i] <= '9') THEN  hasil = (hasil \* 10) + (masukkan[i] - 48)  i++  ELSE THEN  salah = 1  masukkan[i] = '\0'  END IF  END WHILE  IF(salah == 1) THEN  write "format anda salah, masukkan format dengan benar: "  RETURN validasiNim()  ELSE IF(i != 10) THEN  write "NIM harus 10 digit, masukkan NIM dengan benar: "  RETURN validasiNim()  ELSE THEN  RETURN hasil  END IF  END  FUNCTION validasiNama(char masukkan[]): int  DEKLARASI:  i = 0: int  salah = 0: int  DESKRIPSI:  BEGIN  WHILE(masukkan[i] != '\0') THEN  IF((masukkan[i] >= 65 && masukkan[i] <= 90) || (masukkan[i] >= 97 && masukkan[i] <= 122) || masukkan[i] == ' ' || masukkan[i] == '.' || masukkan[i] == ',') THEN  i++  ELSE THEN  salah = 1  masukkan[i] = '\0'  END IF  END WHILE    IF(salah == 1) THEN  RETURN 0  ELSE THEN  RETURN 1  END IF  END  FUNCTION cekNIM(struct mahasiswa data, struct mahasiswa dataTmp[], int jumlah): int  DEKLARASI:  i: int  ditemukan = 0: int  DESKRIPSI:  BEGIN  FOR(i = 0; i < jumlah, i++) THEN  IF(data.nim == dataTmp[i].nim) THEN  ditemukan = 1  BREAK  ELSE THEN  ditemukan = 0  END IF  END FOR    RETURN ditemukan  END  FUNCTION cekNimEdit(struct mahasiswa data, struct mahasiswa dataTmp, int jumlah, int nomorEdit)  DEKLARASI:  i: int  ditemukan = 0: int  DESKRIPSI:  BEGIN  FOR(i = 0; i < jumlah; i++) THEN  IF((data.nim == dataTmp[i].nim) && (dataTmp[i].nim != dataTmp[nomorEdit].nim)) THEN  ditemukan = 1  BREAK  ELSE THEN  ditemukan = 0  END IF  END FOR    RETURN ditemukan  END  FUNCTION prosesReadDataTmp(struct mahasiswa dataTmp[]): int  DEKLARASI:  jumlah = 0: int  DESKRIPSI:  BEGIN  FILE \*readTmp = fopen("dataMahasiswa.txt","r")  IF(!readTmp) THEN  jumlah = 1  ELSE THEN  WHILE(!feof(readTmp)) THEN  fscanf(readTmp,"%d#%[^\n]", &dataTmp[jumlah].nim, &dataTmp[jumlah].nama)  jumlah++  END WHILE  END IF  fclose(readTmp)  RETURN jumlah  END  FUNCTION prosesTambahData(struct mahasiswa data): void  DESKRIPSI:  BEGIN  FILE \*tambah = fopen("dataMahasiswa.txt","a")    fprintf(tambah,"%lu#%s", data.nim, data.nama)  fclose(tambah)  END  FUNCTION prosesLihatData(struct mahasiswa dataTmp[], int jumlah): void  DEKLARASI:  i: int  DESKRIPSI:  BEGIN  IF(jumlah == 1 && dataTmp[0].nim == 0) THEN  write "tidak ada data mahasiswa"  ELSE THEN  FOR(i = 0; i < jumlah; i++) THEN  write "%d. %lu - %s", i+1, dataTmp[i].nim, dataTmp[i].nama  END FOR  END IF  END  FUNCTION prosesPerbaharuiData(struct mahasiswa dataEdit, struct mahasiswa dataTmp[], int jumlah, int nomorEdit): void  DEKLARASI:  i, j: int  DESKRIPSI:  BEGIN  dataTmp[nomorEdit].nim = dataEdit.nim  FOR(i = 0; i < 50; i++) THEN  dataTmp[nomorEdit].nama[i] = dataEdit.nama[i]  END FOR  FILE \*perbaharui = fopen("dataMahasiswa.txt","w")  FOR(i = 0; i < jumlah; i++) THEN  fprintf(perbaharui, "%lu#%s", dataTmp[i].nim, dataTmp[i].nama)  END FOR  fclose(perbaharui)  END  FUNCTION prosesHapusData(struct mahasiswa dataTmp[], int jumlah, int nomorHapus): void  DEKLARASI:  i: int  DESKRIPSI:  BEGIN  FILE \*hapus = fopen("dataMahasiswa.txt","w")  FOR(i = 0; i < jumlah; i++) THEN  IF(i == nomorHapus) THEN  continue  ELSE THEN  fprintf(hapus, "%lu#%s", dataTmp[i].nim, dataTmp[i].nama)  END IF  END FOR  fclose(hapus)  END  FUNCTION tambahData(): void  DEKLARASI:  statusNim: int  statusNama: int  mahasiswa data: struct  mahasiswa dataTmp[100]: struct  DESKRIPSI:  BEGIN  setDefaultDataTmp(dataTmp)  int jumlah = prosesReadDataTmp(dataTmp)  write "tambah data, masukkan nim: "  data.nim = validasiNim()  statusNim = cekNim(data, dataTmp, jumlah)  WHILE(statusNim == 1) THEN  write "nama sudah ada dalam database, masukkan nim yang berbeda: "  data.nim = validasiNim()  statusNim = cekNim(data, dataTmp, jumlah)  END WHILE  write "masukkan nama: "  read(data.nama)  fflush(stdin)  statusNama = validasiNama(data.nama)  WHILE(statusNama == 0) THEN  write "format nama salah, masukkan nama dengan benar: "  read(data.nama)  fflush(stdin)  statusNama = validasiNama(data.nama)  END WHILE  prosesTambahData(data)    write "berikut data yang sudah diinputkan"  write "nim: %lu, data.nim"  write "nama: %s, data.nama"  write "data sudah berhasil ditambahkan"  pilihanUlang(1)  END  FUNCTION lihatData(): void  DEKLARASI:  mahasiswa dataTmp[100]: strcut  DESKRIPSI:  BEGIN  setDefaultDataTmp(dataTmp)  int jumlah = prosesReadDataTmp(dataTmp)  write "lihat data mahasiswa"    prosesLihatData(dataTmp, jumlah)  IF(jumlah != 1 || dataTmp[0].nim != 0) THEN  write "jumlah mahasiswa: %d", jumlah  END IF  pilihanUlang(2)  END  FUNCTION perbaharuiData(): void  DEKLARASI:  cekNimEditStatusNim, statusNim, statusNama: int  mahasiswa dataEdit: struct  mahasiswa dataTmp[100]: struct  DESKRIPSI:  BEGIN  setDefaultDataTmp(dataTmp)  int jumlah = prosesReadDataTmp(dataTmp)  int nomorEdit = 0  write "perbaharui data, daftar data mahasiswa: "  prosesLihatData(dataTmp, jumlah)  IF(jumlah != 1 || dataTmp[0].nim != 0) THEN  write "pilih yang ingin diperbaharui (1-%d): ", jumlah  nomorEdit = input(0, 0, jumlah)  nomotEdit = nomorEdit - 1  write "input perbaharuan data"    write "masukkan nim: "  dataEdit.nim = validasiNim()  statusNim = cekNimEditStatusNim == (dataEdit, dataTmp, jumlah, nomorEdit)  WHILE(statusNim == 1) THEN  write "nim masih sama, masukkan nim yang berbeda: "  dataEdit.nim = validasiNim()  statusNim = cekNimEdit(dataEdit, dataTmp, jumlah, nomorEdit)  END WHILE  write "masukkan nama: "  read(dataEdit.nama)  fflush(stdin)  statusNama = validasiNama(dataEdit.nama)  WHILE(statusNama == 0) THEN  write "format nama salah, masukkan nama dengan benar: "  read(dataEdit.nama)  fflush(stdin)  statusNama = validasiNama(dataEdit.nama)  END WHILE  prosesPerbaharuiData(dataEdit, dataTmp, jumlah, nomorEdit)  write "data berhasil diperbaharui"    ELSE IF(jumlah == 1 && dataTmp[0].nim == 0) THEN  write "masukkan data terlebih dahulu"  END IF  pilihanUlang(3)  END  FUNCTION hapusData(): void  DEKLARASI:  mahasiswa dataTmp[100}: struct  DESKRIPSI  BEGIN  setDefaultDataTmp(dataTmp)  int jumlah = prosesReadDataTmp(dataTmp)  int nomorHapus = 0  int yakinHapus = 0  write "hapus data, data mahasiswa: "    prosesLihatData(dataTmp, jumlah)  IF(jumlah != 1 || dataTmp[0].nim != 0) THEN  write "pilih data yang akan dihapus (1-%d): ", jumlah  nomorHapus = input(0, 0, jumlah)  nomorHapus = nomorHapus - 1  yakinHapus = pertanyaanHapus()  IF(yakinHapus == 1) THEN  prosesHapusData(dataTmp, jumlah, nomorHapus)  write "data berhasil dihapus"  ELSE THEN  write "data tidak berhasil dihapus"  END IF  ELSE IF(jumlah == 1 && dataTmp[0].nim == 0) THEN  write "masukkan data terlebih dahulu"  END IF  pilihanUlang(4)  END  FUNCTION main(): int  DESKIRPSI:  BEGIN  system(cls)  write "nama kelompok"  system(pause)  system(cls)  menuUtama()  RETURN 0  END  FUNCTION menuUtama(): void  DEKLARASI:  pilihan: int  DESKRIPSI:  BEGIN  write "menu 1. tambah data, 2. lihat data, 3. update data, 4. hapus data, 5. keluar"  write "pilihan: "  pilihan = input(0, 0, 5)  IF(pilihan == 1) THEN  tambahData()  ELSE IF(pilihan == 2) THEN  lihatData()  ELSE IF(pilihan == 3) THEN  perbaharuiData()  ELSE IF(pilihan == 4) THEN  hapusData()  ELSE IF(pilihan == 5) THEN  keluar()  END IF  END  FUNCTION keluar(): void  DESKRIPSI:  BEGIN  system(cls)  write "terima kasih"  system(exit)  END |

Kode Program 8.4 *Pseudocode* Program Data Mahasiswa

*Pseudocode* pada Kode Program 8.4 akan membantu *programmer* dalam membuat program yang sebenarnya. *Pseudocode* ini sudah dapat diimplementasikan ke dalam sebuah program yang sudah siap digunakan dengan menambah sintaks-sintaks program yang kurang.

#### Trace Program Data Mahasiswa

*Trace* program adalah suatu proses untuk menemukan *input* yang dimasukkan oleh *user*, dan menelaah hasil *input* tersebut, serta memahami cara pemrosesan yang ada pada suatu program untuk mendapatkan *output* yang sesuai. Di bawah ini adalah program data mahasiswayang dijabarkan dalam bentuk tabel berikut.

Tabel 8.4 *Trace* Program Data Mahasiswa

|  |  |
| --- | --- |
| *Input* | pilihan = 1  masukkan nim = 2205551001  masukkan nama = budi |
| Proses | FILE \*tambah = fopen(“dataMahasiswa.txt”,‘’a’’)  fprint(tambah, data.nim, data.nama)  fclose(tambah) |
| *Output* | print “ini input yang sudah dilakukan”  print “nim = 2205551001”  print “nama = budi”  print “data berhasil ditambahkan”  print “apakah ingin mengulang proses? (y = iya / b = menu / o = keluar): “ |

Tabel 8.4 merupakan hasil *trace* dari program data mahasiswa. Pada saat menjalankan program, *user* akan memasukkan pilihan menu. Jika pilihan, yaitu menu tambah data, maka masukkan NIM dan nama mahasiswa. Selanjutnya, akan masuk ke dalam tahap pemrosesan sesuai dengan program yang sudah ditentukan, serta kemudian akan menampilkan *output* hasil dari proses tersebut.

### Flowchart, Pseudocode, dan Trace Program Asisten

Program asisten ini mampu menyimpan data nilai mahasiswa ke dalam *file* asistenin.txtdan melakukan *sorting* pada nilai tersebut untuk menentukan *ranking* mahasiswa dan menyimpannya pada *file* asistenout.txt. Berikut adalah *flowchart, pseudocode,* serta *trace* dari program asisten.

#### Flowchart Program Asisten

*Flowchart* merupakan suatu diagram yang tersusun atas simbol-simbol serta tanda panah sebagai penanda dari proses dan alur yang berjalan dalam suatu algoritma. Berikut ini merupakan *flowchart* dari program asisten.

##### Flowchart Int Main() dan MenuAwal()

Fungsi utama yang akan dieksekusi oleh bahasa pemrograman C adalah *int main(),* yang juga menandakan awal dan akhir dari suatu program*.* Pada program ini, prosedur pertama yang akan dipanggil oleh *int main()* merupakan prosedur *menuAwal()*. Berikut ini merupakan *flowchart* dari *int main()* beserta *menuAwal()*.

|  |
| --- |
|  |

Gambar 8.38 *Flowchart* Fungsi *Int Main()* dan Prosedur *MenuAwal()*

Gambar 8.38 merupakan *flowchart* dari fungsi *int main()* serta prosedur *menuAwal().* Fungsi *int main()* akan memanggil prosedur *menuAwal()*, yang mana akan menampilkan *output* berupa instruksi untuk pengguna agar memilih menu yang diinginkan. Prosedur ini akan mengarahkan pengguna ke menu selanjutnya (memanggil prosedur tertentu) dengan membandingkan *input* dari pengguna dengan kondisi dari *if-else* yang disediakan.

##### Flowchart Prosedur InputData() dan AsistenIn()

Bilamana pengguna memberikan *input* 1 (tambah data ke *file* asistenin.txt), maka program akan memanggil prosedur *asistenIn()* dengan prosedur *inputData()* di dalamnya. Berikut ini adalah *flowchart* dari prosedur *inputData()* dan *asistenIn()*.

|  |
| --- |
|  |

Gambar 8.39 *Flowchart* Prosedur *InputData()* dan *AsistenIn()*

Gambar 8.39 merupakan *flowchart* dari prosedur *inputrData()* dan *asistenIn().* Pada prosedur *asistenIn()*, pengguna dapat menambahkan data nama berserta nilai dari mahasiswa tersebut ke dalam *file* asistenin.txt. Demi mencegah data duplikat, prosedur ini akan membuka atau membuat *file* asistenin.txt dengan mode *append+* untuk memeriksa apakah nama yang ingin dimasukkan oleh pengguna sudah ada atau belum. *File* ini akan ditutup dan dibuka kembali dengan mode *append* untuk menulis data yang di-*input* sebelumnya oleh pengguna ke dalam *file*. Pengguna dapat meng-*input* lebih dari satu data atau kembali ke menu.

##### Flowchart Prosedur HapusData()

Pada menu kedua, pengguna dapat memilih untuk menghapus data mahasiswa yang sudah tertulis sebelumnya melalui prosedur *hapusData()*. Berikut ini adalah *flowchart* dari prosedur *hapusData()*.

|  |
| --- |
|  |

Gambar 8.40 *Flowchart* Prosedur *HapusData()*

Gambar 8.40 merupakan *flowchart* dari prosedur *hapusData().* Pada prosedur ini, pengguna dapat menghapus data yang sudah dituliskan sebelumnya ke dalam *file asistenin.txt.* Bilamana ternyata *file* tidak ditemukan, maka program akan memberikan instruksi bahwa belum ada masukan data dan memanggil prosedur *menuAwal().* Bila data yang ingin dihapus tidak ditemukan, maka program akan memberikan instruksi bahwa data tidak ditemukan dan memanggil prosedur *repeat()*. Bila ditemukan, maka program akan membuat *file tempAsis.txt* untuk menyimpan data baru tanpa data yang ingin dihapus, lalu mengganti *file asistenin.txt* dengan *file tempAsis.txt* dan mengganti nama *file tempAsis.txt* menjadi *asistenin.txt*.

##### Flowchart Prosedur AsistenOut()

Bila pengguna memilih untuk mengurutkan data mahasiswa berdasarkan nilai, maka program akan memanggil prosedur *asistenOut()*. Berikut ini adalah *flowchart* dari prosedur *asistenOut()*.

|  |
| --- |
|  |

Gambar 8.41 *Flowchart* Prosedur *AsistenOut()*

Gambar 8.41 merupakan *flowchart* dari prosedur *asistenOut()*. Dalam prosedur ini, data yang ada pada *file asistenin.txt* akan diurutkan berdasarkan nilai terbesar dan hasilnya ditampilkan sebagai *output* pada layar serta ditulis dalam *file asistenOut.txt*. Bilamana *file asistenin.txt* tidak ditemukan, maka program akan memberikan instruksi bahwa belum ada masukan data dan memanggil prosedur *menuUtama().*

##### Flowchart Prosedur Repeat(), Kredit(), serta Keluar()

*Flowchart* berikutnya adalah prosedur *repeat(), kredit(),* serta *keluar().* Prosedur *repeat()* dan *keluar()* digunakan untuk mempermudah penggunaan program, sementara prosedur *kredit()* digunakan untuk memunculkan nama kelompok. Berikut ini adalah *flowchart* dari ketiga prosedur tersebut.

|  |
| --- |
|  |

Gambar 8.42 *Flowchart* Prosedur *Repeat(), Kredit(),* serta *Keluar()*

Gambar 8.42 merupakan *flowchart* dari prosedur *repeat(), kredit(),* serta *keluar().* Prosedur *repeat()* akan memberikan instruksi pengguna mengenai pilihan untuk kembali ke menu utama atau keluar dan menggunakan *if-else* dalam memproses *input* pengguna setelahnya. Prosedur tersebut dapat ditemukan pada akhir fitur program. Prosedur *keluar()* akan dipanggil bila pengguna memilih untuk keluar dari program, yang mana akan menghentikan program. Prosedur *kredit()* ditampilkan untuk menunjukkan kredit atau nama kelompok dan selanjutnya akan kembali ke menu awal.

#### Pseudocode Program Asisten

Salah satu notasi algoritma yang kerap digunakan selain *flowchart* adalah *pseudocode*, yang mana menggambarkan suatu alur algoritma dalam bentuk yang mirip seperti kode program, hanya saja lebih mudah dimengerti oleh manusia dan tidak terikat dengan *syntax* bahasa pemrograman mana pun. Berikut ini adalah *pseudocode* dari program asisten.

|  |
| --- |
| PROGRAM asistensi  BEGIN  DECLARE  TYPEDEF STRUCT {  char nama[30]  float nilai  } dataMahasiswa  dataMahasiswa write, read, data[30]  FUNCTION validasiFloat(int \*var)  BEGIN  fflush(stdin)  DECLARE  char buffer[1024]  char cek  IF (fgets(buffer, sizeof(buffer), stdin) != NULL)  IF (sscanf(buffer, "%f %c", var, &cek) == 1)  RETURN \*var  RETURN IF  END IF  RETURN 0  ENDFUNCTION  PROCEDURE validasiInput(float \*var)  BEGIN  WHILE (1)  fflush(stdin)  IF CALL (validasiFloat(var))  BREAK  END IF  WRITE "Masukan salah. Mohon masukkan bilangan yang sesuai."  WRITE "Masukkan input: "  ENDWHILE  ENDPROCEDURE  PROCEDURE validasiRange(float \*var, int batas1, int batas2)  BEGIN  WHILE (1)  CALL validasiInput(var)  fflush(stdin)  IF (batas1 <= \*var && \*var <= batas2)  BREAK  WRITE "Masukan salah. Mohon masukkan bilangan dengan range yang sesuai."  WRITE "Masukkan input: "  ENDWHILE  ENDPROCEDURE  PROCEDURE inputData()  BEGIN  WRITE "Masukkan nama mahasiswa: "  READ fgets(write.nama, 30 , stdin)  write.nama[strcspn(write.nama, "\n")] = '\0'  fflush(stdin)  WRITE "Masukkan nilai mahasiswa: "  CALL validasiRange(&write.nilai, 0, 100)  ENDPROCEDURE  PROCEDURE asistenIn()  BEGIN  DECLARE  float pilihan  WRITE "INPUT DATA NILAI MAHASISWA"  WRITE "Mohon masukkan nama dan nilai mahasiswa."  WRITE "Data ini akan tersimpan pada file asistenin.txt"  WHILE (1)  CALL inputData()  DECLARE  FILE \*cekInput  cekInput = OPENFILE "asistenin.txt" FOR APPEND+  WHILE (!feof(cekInput))  READFILE "asistenin.txt", "%[^\n]\n%f\n", &read.nama, &read.nilai  IF (strstr(read.nama, write.nama)!= 0)  WRITE "[PERINGATAN]"  WRITE "Tidak dapat memasukkan data yang sama."  WRITE "Data yang sudah ada:"  WRITE "Nama mahasiswa: %s Nilai mahasiswa: %.2f\n\n", read.nama, read.nilai  WRITE "Mohon masukkan data yang baru."    CALL inputData()  ENDIF  ENDWHILE  CLOSEFILE "asistenin.txt"  DECLARE  FILE \*asistenin  asistenin = OPENFILE "asistenin.txt" FOR APPEND  WRITEFILE "asistenin.txt", "%s\n%.2f\n", write.nama, write.nilai  CLOSEFILE "asistenin.txt"  WRITE "Data telah ditambahkan."  WRITE "Ingin memasukkan data lagi? [1] Ya [2] Tidak"  CALL validasiRange(&pilihan, 1, 2)  IF (pilihan == 2) BREAK  ENDIF  ENDWHILE  ENDPROCEDURE  PROCEDURE asistenOut()  BEGIN  WRITE "MENGURUTKAN RANKING NILAI MAHASISWA"  WRITE "Di bawah ini adalah data nilai mahasiswa yang sudah diurutkan."  WRITE "Data ini akan tersimpan pada file asistenout.txt"    DECLARE  FILE \*asistenin  asistenin = OPENFILE "asistenin.txt" FOR READING  IF asistenin == NULL  WRITE "Belum ada masukan data. Silahkan masukkan data terlebih dahulu."  system("pause")  system("cls")  CALL menuAwal()  ELSE rewind(asistenin)  ENDIF  DECLARE  int i = 0  WHILE (!feof(asistenin))  READFILE "asistenin.txt", "%[^\n]\n%f\n", &data[i].nama, &data[i].nilai)  i++  ENDWHILE  CLOSEFILE "asistenin.txt"  DECLARE  int n, j, tempNilai  char tempNama[30]  FOR (n = 0 ; n < i - 1; n++)  FOR (j = 0 ; j < i - n - 1 ; j++)  IF (data[j].nilai < data[j+1].nilai)  tempNilai = data[j].nilai  data[j].nilai = data[j+1].nilai  data[j+1].nilai = tempNilai  strcpy(tempNama, data[n].nama)  strcpy(data[n].nama, data[j].nama)  strcpy(data[j].nama, tempNama)  ENDIF  ENDFOR  ENDFOR    WRITE " NO | Nama Mahasiswa | Nilai "  FOR (n = 0 ; n < i; n++)  WRITE " %-2d.| %-51s | %.2f ", n+1, data[n].nama, data[n].nilai  ENDFOR  DECLARE  FILE \*asistenout  asistenout = OPENFILE "asistenout.txt", FOR WRITING  WRITEFILE "asistenout.txt", "DAFTAR RANKING MAHASISWA"  WRITEFILE "asistenout.txt", " NO | Nama Mahasiswa | Nilai"    FOR (n = 0 ; n < i; n++)  WRITEFILE "asistenout.txt", " %-2d.| %-51s | %.2f ", n+1, data[n].nama, data[n].nilai  ENDFOR  CLOSEFILE "asistenout.txt"  ENDPROCEDURE  PROCEDURE hapusData()  BEGIN  WRITE "MENGHAPUS DATA NILAI MAHASISWA"  WRITE "Pilih data nilai mahasiswa yang ingin dihapus"  WRITE "Aksi ini akan mempengaruhi isi dari asistenin.txt"    DECLARE  FILE \*asistenin  asistenin = OPENFILE "asistenin.txt", FOR READING  fseek(asistenin, 0, SEEK\_END)  IF (ftell(asistenin)==0){  WRITE "Belum ada masukan data. Silahkan masukkan data terlebih dahulu."  system("pause")  system("cls")  CALL menuAwal()  ELSE rewind(asistenin)  ENDIF  WRITE "NO | Nama Mahasiswa | Nilai"  DECLARE  int print = 1  WHILE (!feof(asistenin))  READFILE "asistenin.txt" "%[^\n]\n%d\n", &read.nama, &read.nilai  WRITE " %-2d.| %-51s | %.2f ", print, read.nama, read.nilai  print++  ENDWHILE    WRITE "Ketik nama yang ingin dihapus: "  READ fgets(write.nama, 30 , stdin)  write.nama[strcspn(write.nama, "\n")] = '\0'  fflush(stdin)  rewind(asistenin)  DECLARE  int cari=0;  WHILE (!feof(asistenin))  READFILE "asistenin.txt" "%[^\n]\n%f\n", &read.nama, &read.nilai)  IF (strstr(read.nama, write.nama)!= 0)  cari=1  BREAK  ELSE cari=0  ENDIF  ENDWHILE  IF (cari==0)  WRITE "Data tidak ditemukan."  CALL repeat()  ENDIF  DECLARE  FILE \*tempAsis  tempAsis= OPENFILE "tempAsis.txt" FOR WRITING  rewind(asistenin)  WHILE (!feof(asistenin))  READFILE "asistenin.txt" "%[^\n]\n%f\n", &read.nama, &read.nilai  IF (strstr(read.nama, write.nama)== 0)  WRITEFILE "%s\n%.2f\n", read.nama, read.nilai  ENDIF  ENDWHILE  WRITE "Data telah dihapus dari file."  CLOSEFILE "asistenin.txt"  CLOSEFILE "tempAsis.txt"  REMOVE "asistenin.txt"  RENAME "tempAsis.txt","asistenin.txt"  ENDPROCEDURE  PROCEDURE repeat()  BEGIN  DECLARE  float pilihanUlang;  WRITE "Apakah Anda ingin kembali ke menu utama?"  WRITE "Ketik [1] untuk 'Ya' dan [2] untuk 'Tidak' :"  CALL validasiMenu(&pilihanUlang, 1, 2)  IF (pilihanUlang==1)  system("cls")  CALL menuAwal()  ELSEIF (pilihanUlang==2){  system("cls")  keluar()  ENDIF  ENDPROCEDURE  PROCEDURE kredit()  BEGIN  WRITE "K E L O M P O K 2 3"  WRITE "PROGRAM INPUT DAN RANKING NILAI MAHASISWA"  WRITE "Nama Anggota Kelompok"  WRITE "205551005 | Gede Made Rapriananta Pande"  WRITE "2205551003 | Ida Bagus Paalakaa RNB"  WRITE "2205551069 | Ni Kadek Ari Diah Lestari"  WRITE "2205551072 | I Gusti Ayu Krisna Kusuma Dewi"  WRITE "2205551076 | Kadek Yogi Dwi Putra Utama"  WRITE "2205551079 | Anak Agung Indi Kusuma Putra"    system("pause")  system("cls")  CALL menuAwal()  ENDPROCEDURE  PROCEDURE keluar()  BEGIN  WRITE "TERIMA KASIH"  WRITE "Sampai jumpa di lain waktu!"  exit(0)  ENDPROCEDURE  PROCEDURE menuAwal()  BEGIN  DECLARE  int pilihan  WRITE "PROGRAM INPUT DAN RANKING NILAI MAHASISWA"  WRITE "Memasukkan Data Nilai dan Mengurutkan Ranking"  WRITE "Pilih Menu yang Diinginkan"  WRITE "[1] Tambah Data Baru (asistenin.txt)"  WRITE "[2] Hapus Data"  WRITE "[3] Urutkan Data (asistenout.txt)"  WRITE "Menu Lainnya"  WRITE "[4] Kredit"  WRITE "[5] Keluar dari program"  WRITE "Pilih menu yang diinginkan: "  CALL validasiRange(&pilihan, 1, 5)  CALL validasiRange(&pilihan, 1, 5)  IF (pilihan == 1)  system("cls")  CALL asistenIn()  ELSEIF (pilihan == 2)  system("cls")  CALL hapusData()  ELSEIF (pilihan == 3)  system("cls")  CALL asistenOut()  ELSEIF (pilihan == 4)  CALL system("cls")  CALL kredit()  ELSEIF (pilihan == 5)  CALL system("cls")  CALL keluar()  ENDIF  CALL repeat()  ENDPROCEDURE  int main(void)  {  menuAwal()  return 0  } |

Kode Program 8. 5 *Pseudocode* Program Asisten

Kode Program 8.5 merupakan *pseudocode* dari program asisten. Setelah merancang *pseudocode* tersebut, *programmer* pun akan mampu mengimplementasikan algoritma yang sesuai ke dalam kode program yang asli dengan lebih mudah atau pun mempermudah komunikasi antar *programmer* mengenai kode program yang akan dibuat dengan berpacu pada *pseudocode* yang telah disusun.

#### Trace Program Asisten

*Trace* merupakan metode yang dilaksanakan untuk memeriksa apakah alur algoritma yang dipakai sudah benar atau belum. *Trace* atau pelacakan dilakukan dari *input,* lalu dilanjutkan dengan proses yang dilakukan berdasarkan *input* tersebut, dan diakhiri dengan *output* hasil dari proses tersebut. Berikut adalah *trace* dari program asistensi.

Tabel 8.5 *Trace* Program Asisten

|  |  |
| --- | --- |
| *Input* | pilihan = 1  write.nama = Olive  write.nilai = 78  pilihan = 1  write.nama = Tuna  write.nilai = 98  pilihan = 1  write.nama = Krista  write.nilai = 54.76  pilihan = 2  pilihanUlang = 1  pilihan = 3  pilihanUlang = 1  pilihan = 2  write.nama = Tuna  pilihanUlang = 2 |
| Proses | CALL menuAwal()  CALL asistenIn()  CALL repeat()  CALL menuAwal()  CALL asistenOut()  CALL repeat()  CALL menuAwal()  CALL hapusData()  CALL repeat()  CALL keluar() |
| *Output* | Pilihan menu  Data telah ditambahkan  Ingin memasukkan data lagi?  Data telah ditambahkan  Ingin memasukkan data lagi?  Data telah ditambahkan  Ingin memasukkan data lagi?  Ingin kembali ke menu utama?  Pilihan menu  Output Ranking  1. Tuna 98.00  2. Olive 78.00  3. Krista 54.76  Ingin kembali ke menu utama?  Pilihan menu  Output daftar data mahasiswa  Instruksi ketik nama yang ingin dihapus  Data telah dihapus dari file  Ingin kembali ke menu utama?  Terima kasih |

Tabel 8.5 di atas merupakan contoh dari pelaksanaan metode *trace* untuk memeriksa apakah algoritma yang dipakai sudah tepat atau belum. Pada *trace* tersebut, pengguna memilih untuk menambahkan data pada *file asistenin.txt* dan lalu memasukkan data mahasiswa yang bersangkutan sebanyak tiga kali. Data tersebut lantas disimpan pada *file* *asistenin.txt*. Proses ini terjadi pada prosedur asistenIn().

Setelahnya, pengguna ingin mengurutkan ranking dari mahasiswa tersebut dan memilih menu untuk mengurutkan data. Program akan memanggil prosedur asistenOut(), yang mana akan mengurutkan data mahasiswa, menampilkan hasilnya di layar, serta menyimpan hasil pengurutan pada *file asistenout.txt*.

Pengguna lantas kembali ke menu utama dan beranjak ke menu untuk menghapus data. Program pun memanggil prosedur hapusData() untuk melaksanakannya. Setelah pengguna memberikan *input* berupa nama mahasiswa yang ingin dihapus, program pun akan mengganti *file asistenin.txt* dengan file *asistenin.txt* yang baru, yang mana dalam *file* tersebut, data yang ingin dihapus tidak dimasukkan. Pengguna pun mengakhiri program tersebut.

# Daftar Pustaka

Bartwal, U., 2022. *What is data type of FILE in C ?.* [Online]   
Available at: https://www.geeksforgeeks.org/data-type-file-c/  
[Diakses 16 November 2022].

Bryant, R. E. & O'Hallaron, D. R., 2016. *Computer Systems A Programmer's Perspective.* 2016 penyunt. Malaysia: Pearson Education Limited.

Deepali, 2022. *Understanding the Difference Between Structure and Union in C.* [Online]   
Available at: https://www.naukri.com/learning/articles/difference-between-structure-and-union-in-c/  
[Diakses 15 November 2022].

Educative, 2022. *Splitting a string using strtok() in C.* [Online]   
Available at: https://www.educative.io/answers/splitting-a-string-using-strtok-in-c  
[Diakses 15 November 2022].

GeeksforGeeks, 2019. *Difference between Public and Private in C++ with Example.* [Online]   
Available at: https://www.geeksforgeeks.org/difference-between-public-and-private-in-c-with-example/  
[Diakses 14 November 2022].

GeeksforGeeks, 2021. *fgetc() and fputc() in C.* [Online]   
Available at: https://www.geeksforgeeks.org/fgetc-fputc-c/  
[Diakses 16 November 2022].

GeeksforGeeks, 2022. *Basics of File Handling in C.* [Online]   
Available at: https://www.geeksforgeeks.org/basics-file-handling-c/  
[Diakses 16 November 2022].

GeeksforGeeks, 2022. *Write your own atoi().* [Online]   
Available at: https://www.geeksforgeeks.org/write-your-own-atoi/#:~:text=The%20atoi()%20function%20in,string%20argument%20to%20an%20integer.  
[Diakses 15 November 2022].

Goyal, Y., 2022. *fseek() in C.* [Online]   
Available at: https://www.educba.com/fseek-in-c/  
[Diakses 20 November 2022].

Hanly, J. R. & Koffman, E. B., 2013. *Problem Solving and Program Design in C.* 7th penyunt. New Jersey: Pearson Education, Inc.

IBM, 2021. *atof() — Convert Character String to Float.* [Online]   
Available at: https://www.ibm.com/docs/en/i/7.4?topic=functions-atof-convert-character-string-float#itof  
[Diakses 15 November 2022].

IBM, 2021. *atoi() — Convert Character String to Integer.* [Online]   
Available at: https://www.ibm.com/docs/en/i/7.4?topic=functions-atoi-convert-character-string-integer  
[Diakses 15 November 2022].

Kalicharan, N., 2015. *Learn to Program with C.* New York: Springer Science+Business.

Kumar, S., 2022. *How to write in a file using fputs() in C.* [Online]   
Available at: https://www.geeksforgeeks.org/how-to-write-in-a-file-using-fputs-in-c/  
[Diakses 16 November 2022].

Pedamkar, P., 2022. *Access Modifiers in C++.* [Online]   
Available at: https://www.educba.com/access-modifiers-in-c-plus-plus/  
[Diakses 15 November 2022].

Thompson, B., 2022. *C Files I/O: Create, Open, Read, Write and Close a File.* [Online]   
Available at: https://www.guru99.com/c-file-input-output.html  
[Diakses 16 November 2022].

Thompson, B., 2022. *Strings in C: How to Declare & Initialize a String Variables in C.* [Online]   
Available at: https://www.guru99.com/c-strings.html  
[Diakses 15 November 2022].

w3chools, 2022. *C++ Access Specifiers.* [Online]   
Available at: https://www.w3schools.com/cpp/cpp\_access\_specifiers.asp#:~:text=public%20%2D%20members%20are%20accessible%20from,viewed)%20from%20outside%20the%20class  
[Diakses 12 November 2022].